(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 413 132 A2**

(12) **EUROPEAN PATENT APPLICATION**

(21) Application number: **90113224.1**

(51) Int. Cl.5: **G06F 15/38**

(22) Date of filing: **11.07.90**

(30) Priority: **16.08.89 US 395118**

(43) Date of publication of application:
20.02.91 Bulletin 91/08

(84) Designated Contracting States:
DE FR GB

(71) Applicant: **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: **Jensen, Karen**
5822 Inman Park Circle, No. 210
Rockville, MD 20852(US)

(74) Representative: **Jost, Ottokarl, Dipl.-Ing.**
IBM Deutschland GmbH Patentwesen und Urheberrecht Schönaicher Strasse 220
D-7030 Böblingen(DE)

(54) A computer method for identifying predicate-argument structures in natural language text.

(57) A computer method is disclosed for determining predicate-argument structures in input prose sentences of English. The input sentence, in the form of a string of words separated by blanks, is first analyzed (parsed) by a rule component that has access only to morphological and syntactic information about the words. The output of this rule component, in the form of a data structure consisting of attribute-value pairs, is then processed by the argument-structure component, which consists of a set of partially ordered procedures that incorporate further linguistic knowledge. The output of these procedures is the same attribute-value structure, now enhanced by the presence of semantic (i.e., meaningful, non-syntactic) attributes. These semantic attributes, taken together, form the argument structure of the input sentence.

The resulting invention constitutes a fully modular, comprehensive and efficient method for passing from syntax to the first stage of semantic processing of natural (human) language. The invention applies to all prose sentences of the language for which it is designed, and not just to a subset of those sentences. It does not use domain-specific semantic information to improve the accuracy or efficiency of the syntactic component. It therefore constitutes an unrestricted broad-coverage method for natural language processing (NLP), as opposed to the restricted methods used in most NLP applications today.

Although the specific rules and procedures will be different for different natural languages, the general concept embodied in this invention is applicable to all natural languages.

FIG. 3

```
SYNTACTIC PARSE TREE FOR THE SENTENCE:
- - - - - - - - - - - - - - - - - - - - - - - - - -
49 —DECL1  . NP1      NOUN1*     ' ALICE '
                      PUNC1      ' , '
             NP2      NOUN2*     ' PETER '
             VERB1*   ' SAID '
             VP1      COMPL1     ' THAT '
                      NP3        NOUN3*     'JOHN '
                      VERB2*     ' KISSED '
             PUNC2    ' . '
- - - - - - - - - - - - - - - - - - - - - - - - - -

BASIC SEMANTIC ( ARGUMENT ) STRUCTURE FOR THE SENTENCE :

50 —  PRED        ' SAY '
      DSUBJECT    PRED        ' PETER '
      DOBJECT     PRED        ' KISS '
                  DSUBJECT    PRED        ' JOHN '
                  DOBJECT     PRED        ' ALICE '
```

EP 0 413 132 A2

A COMPUTER METHOD FOR IDENTIFYING PREDICATE-ARGUMENT STRUCTURES IN NATURAL LANGUAGE TEXT

The invention disclosed herein broadly relates to data processing methods for natural language processing (NLP), and more particularly relates to an improved data processing method for determining the basic semantic structures of sentences.

Natural language texts may be said to consist of groups of propositions made up of predicates and their arguments. An example of a predicate is a verb, and its arguments can be exemplified by associated nouns or noun phrases. For example, in the sentence:

John loves Mary.

there is one proposition, whose predicate is the verb "loves." "Loves" has two arguments in this proposition: "John" and "Mary." In order for a computer system to understand natural language, it must be able to identify, correctly, the predicate and argument groups. For a simple sentence like the one above, this is not hard. If an English verb is closely surrounded by its arguments (as in "John loves Mary" above), then it is relatively easy for the computer grammar to assign the proper arguments to the verb. But for more complicated sentences, such as many that appear in real-life text, the task becomes much more difficult. The difficult problem arises when the arguments are <u>not</u> close to their verb.

In fact, arguments may sometimes be missing from the sentence entirely, and yet must be inferred by the program, just as a human would infer them. For example:

Mary was kissed.

In this sentence, the only visible argument for the verb "kissed" is "Mary." But we can infer another argument, corresponding to some person who did the kissing. Another, related, situation occurs in sentences like:

Who did Mary think that Peter said that John kissed? In the foregoing sentence, again there are two arguments for the verb "kissed." "John" is close by, but "who," the second argument, is far away from its verb. The problem, then, is properly to link all arguments, including the missing and far-removed ones, with their predicates.

The problem of identifying predicate-argument structures -- and, in particular, of correctly assigning "long-distance dependencies" as in "Who did Mary think that Peter said that John kissed?" -- is well-known in the literature of linguistics and computational linguistics. Two chief methods have been described for accomplishing this:

* the "empty category" (EC) approach;
* the "functional uncertainty" (FU) approach.

The EC approach is advocated, for example, by linguists of the Government and Binding (GB) and Generalized Phrase Structure Grammar (GPSG) schools. (P. Sells, <u>Lectures on Contemporary Syntactic Theories</u> , <u>CSLI, Stanford University, Stanford, CA, 1985.)</u> This approach uses parse structures that contain empty slots in the places where the dislocated constituents might be, if the sentence were in its most neutral form. For example, the sentence Alice, Peter said that John kissed. (= Peter said that John kissed Alice.) is supposed to have an "empty category," or "trace" (symbolized by "e"), right after the verb "kissed," because that is where the noun phrase "Alice" belongs. Computational grammars that are built along these lines actually specify empty slots in their parse structures, or trees (see Fig. 1A).

The FU approach is advocated by linguists who adhere to the theories of Lexical Functional Grammar (LFG). This approach bases its solution not on empty slots in a parse tree, but rather on the incremental evaluation of the characteristics of all the verbs ("characteristics" chiefly refers to the required number and kind of arguments that a verb must have), from left to right in a sentence, in order to find out where the displaced constituent best fits. A formal notational device has been added to the LFG grammar-writing language, for the purpose of computing the properly filled argument structures. (R. M. Kaplan and A. Zaenen, "Long-distance Dependencies, Constituent Structure, and Functional Uncertainty," in M. Baltin and A. Kroch, eds., <u>Alternative Conceptions of Phrase Structure</u> , <u>Chicago University Press, 1987.)</u> Computational grammars that are built along these lines use this device, in their grammar rules, to specify where the missing argument should be assigned.

The present method differs from both of these approaches. It differs from the EC approach in that:

a. It does not use empty categories or traces of any kind;

b. It does not rely so heavily on the constituent, or tree, structure, but rather uses all sorts of information provided by the syntactic parse.

It differs from the FU approach in that:

a. It does not use any special notational devices other than those already provided by the programming language used;

b. It does not rely so completely on characteristics of verbs in the sentence (the so-called "functional information"), but rather uses all sorts of information provided by the syntactic parse.

It differs from both of the above approaches in that it performs the argument-filling after the syntactic parse has been completed. It uses a post-processor, and not the parsing component itself, to manipulate the full range of syntactic attribute-value information, in order to derive the most reasonable argument structure.

An additional difference between the present method and the methods of NLP systems that are motivated by linguistic theories is the fact that most of the latter systems currently use some form of unification, such as that provided by the logic programming languages. Unification allows for an automatic matching of attribute-value structures; but it has several drawbacks, such as its inability to deal elegantly with conditions of negation and disjunction. The present method, using a procedural post-processor, suffers no such drawbacks.

The present method is highly efficient; the post-processor adds no measurable time to the operation of the system. In addition, because the initial parsing component is completely domain-independent, the entire system provides extremely broad coverage for English.

Although the EC approach and the FU approach dominate current linguistic theory, neither one has been widely adopted in applications that make use of NLP techniques today. Prior art applications that include a semantic analysis of English text generally make use of some form of lexically-driven argument identification, but do not necessarily embrace the techniques or formalisms of EC or FU.

A prior art method for semantic processing of English text is disclosed in the Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics, Stanford University, 6-9 July 1987, pp. 131-134. The method disclosed therein is briefly explained below. The prior art system is designed to handle a single semantic domain, namely, reports of failures in a specific type of machinery used on Navy ships. When an English sentence from this domain is inputted, the system makes a syntactic analysis of the sentence, and then maps the syntactic analysis onto an underlying format, or template, that specifies how many arguments can be related to the verb of that sentence, and what sorts of arguments those should be. Three different classes of arguments are defined: (1) obligatory, (2) essential, and (3) non-essential. Obligatory arguments must be present in the syntactic analysis, or the parse fails. Essential arguments need not be present in the syntax; but, if they are not, the system will hypothesize some "best guess" candidate to fill the role. Therefore both the essential and the obligatory arguments end up being present in the semantic structure of the sentence. Non-essential arguments may or may not be present.

For example, given the input sentence "Pump failed," the syntactic analysis should give "failed" as the main verb and "pump" as its syntactic subject. The underlying template for the verb "fail" should indicate that it has one argument, called the PATIENT. A mapping rule then suggests that "pump" is a good candidate for the PATIENT argument (arguments are also called "roles"). Next, restrictions are tested. For the verb "fail," there is a restriction saying that the filler of the PATIENT role must be a mechanical device. (In general, such information is carried by a feature -- say, +MECH -- that is marked on the dictionary entry for the noun "pump.") Since "pump" checks out as a mechanical device, the argument structure is completed: "failed" has one argument, its PATIENT, which is filled by "pump."

However, the prior art argument-filling method has several problems, as discussed below.

First, the possible meanings that words can have are severely limited, including only those that pertain to the domain in question. For example, the verb "fail" can have the meaning associated with sentences like:

The equipment failed.

in which it has one obligatory argument ("equipment"). But the system may not interpret the verb "failed" in sentences like:

His courage failed him.

Today I took the chemistry exam and failed me a whopper!

The system counts on the fact that such sentences usually do not appear within the narrowly defined subdomain. But people use language in unpredictable ways; there is no guarantee that the verb "fail" would never be used, in Navy ship reports, with something like the meanings used above. The only way for the system to handle such sentences would be by means of additional templates for "fail." However, additional templates may cause much trouble for the syntactic analysis component.

Second, the process is complicated by the necessity to separate, for each verb, the three classes of arguments: obligatory, essential, and non-essential. The number of obligatory arguments varies with each different sense of a verb, and it is very difficult to specify precisely how many senses any given verb may have, even within a particular narrow semantic subdomain.

Third, the flow of the system is hampered by the requirement that all essential arguments be filled, even if the filler is only a "best guess" hypothesis. In cases where fewer arguments are present in the syntactic structure than are required by the lists of obligatory and essential arguments, it is often necessary for the system to fail, back up, and try again, before achieving a successful parse for the sentence.

Fourth, in the prior art system, little or no attention is paid to the trickiest kinds of argument-filling, such as the "long-distance dependencies" discussed above. Again, the system counts on the fact that such complicated constructions are not expected to occur in narrow subdomains. Given the flexible nature of natural language, however, this is not a totally safe expectation.

Theoretical approaches to argument-filling discussed above (EC and FU) deal with the complexities of natural language, but their intrinsic complications make them difficult to use in practical applications. Prior art applications, although useable in the real world, within semantic subdomains, do not provide techniques for dealing with the full complexity of natural language, and will therefore remain limited in their scope of application.

Reference is made to USP 4,731,735 to K. W. Borgendale, et al., assigned to IBM Corporation, entitled "Multilingual Processing for Screen Image Build and Command Decode in a Word Processor, With Full Command, Message and Help Support," for its disclosure of a data processing system in which the invention disclosed herein can be executed. The disclosure of the above cited patent is incorporated herein by reference to serve as a background for the invention disclosed herein.

It is therefore an object of the invention to provide an improved method for determining the argument structures, or basic semantic structures, of English sentences.

It is another object of the invention to provide broad coverage of English, so that there is a high probability of success in determining the argument structure for any input sentence of English, and not just for sentences that pertain to a restricted semantic domain.

It is a further object of the invention to provide efficient techniques for semantic processing, by using a fully modular approach coupled with procedures that work entirely by consulting, adding to, and subtracting from, attribute-value data structures, and that are not restricted by any predicate-argument templates that specify how many arguments a verb (or any word) must have in order to be understood.

These and other objects, features and advantages are accomplished by the invention disclosed herein. A computer method is disclosed for determining predicate-argument structures in input prose sentences of English. The input sentence, in the form of a string of words separated by blanks, is first analyzed (parsed) by a rule component that has access only to morphological and syntactic information about the words. The output of this rule component, in the form of a data structure consisting of attribute-value pairs, is then processed by the argument-structure component, which consists of a set of partially ordered procedures that incorporate further linguistic knowledge. The output of these procedures is the same attribute-value structure, now enhanced by the presence of semantic (i.e., meaningful, non-syntactic) attributes. These semantic attributes, taken together, form the argument structure of the input sentence.

The resulting invention constitutes a fully modular, comprehensive and efficient method for passing from syntax to the first stage of semantic processing of natural (human) language. The invention applies to all prose sentences of the language for which it is designed, and not just to a subset of those sentences. It does not use domain-specific semantic information to improve the accuracy or efficiency of the syntactic component. It therefore constitutes an unrestricted broad-coverage method for natural language processing (NLP), as opposed to the restricted methods used in most NLP applications today.

Although the specific rules and procedures will be different for different natural languages, the general concept embodied in this invention is applicable to all natural languages.

A NLP method is disclosed for determining basic semantic structures for English input strings. In order to achieve these objects in accordance with the present method, an input string of English is processed in the following manner. In accordance with the invention, there are two stages in the method, the first being performing a syntactic parsing without a semantic component, and the second being performing the semantic analysis.

First, an input string is analyzed by a syntactic parsing component. As an example, the preferred embodiment uses a syntactic parser called the PLNLP English Grammar (PEG). ("PLNLP" is the name of a programming language; the acronym stands for Programming Language for Natural Language Processing.) PEG provides a syntactic analysis for each input string. The analysis consists of a collection of attributes and values. Such a collection is called a "record" or "record structure."

During the stage of syntactic analysis, PEG makes no attempt to complete the assignment of arguments to each verb. However, the analysis that PEG provides does contain all of the information necessary to figure out, and to fill in, all of the arguments.

The argument structure is created by enhancing the syntactic parse record with additional attributes that are intended to have semantic values. Syntax is often called the "surface" structure, and semantics is called the "deep" structure, of the input string. The following "deep" attributes have been defined; others may be added if needed:

a. DSUBJECT - "deep" (or semantic) subject of the proposition; generally, the doer of an action

b. DOBJECT - "deep" (or semantic) object of the proposition; the entity that is most directly affected by the action of the doer

c. DINDOBJ - deep indirect object; the entity that experiences something, or receives something, through the action of the doer

d. DPREDNOM - the entity that is equated with the DSUBJECT in a proposition

e. DOBJCOMP - the entity that is equated with the DOBJECT in a proposition

After leaving the syntactic parsing stage, in accordance with the invention a record goes to the first step of the semantic analysis stage. Here the semantic arguments are identified in the easy cases -namely, those which are, by syntactic standards, close to and clearly associated with their verbs. This is simply a matter of adding "deep" argument attributes, and making them point to the same values as do their "surface" syntactic counterparts (DSUBJECT = SUBJECT; DOBJECT = OBJECT; etc.)

There are two further steps in the semantic analysis stage. In the second step, arguments are assigned and completed as follows:

* Missing arguments of infinitive clauses and participial clauses are assigned.

* Displaced or "long-distance" arguments are assigned.

* Missing or displaced arguments in passive onstructions are assigned.

* Arguments for the two different forms of the indirect object construction in English are equated.

This completes the step of argument assignment. Optionally, this step can be augmented by devising procedures to handle argument assignment for other syntactic situations, if these are discovered.

Within these core procedures, ordering is important to some degree. For example, the sub-procedure that assigns missing arguments to infinitival and participial clauses must be ordered first in the list. The sub-procedure that handles displaced ("long-distance") arguments must be ordered before the passive procedure.

After all arguments have been properly assigned, the post-processor begins its third step, where it reviews the parsed segments that are <u>not</u> arguments of the main verb. These non-argument segments are called "adjuncts." This review results in the further enhancement of the record structure with the addition of semantic attributes that point to all non-argument modifiers of the major classes VP (verb phrase), AJP (adjective phrase), AVP (adverb phrase), NP (noun phrase), and PP (prepositional phrase). The following semantic attributes have been defined; others may be added when needed:

a. MODS - modifier; not further specified

b. NADJ - adjective pre-modifying noun

c. PADJ - predicate adjective or adjective post-modifying noun

d. OPS - operator; includes determiners and quantifiers

e. PARTICL - preposition or adverb that combines with a verb to signal a significant change in the argument structure of the verb phrase

f. PRED - basic form of each word

g. PROP - propositional modifier; may include infinitives and participial phrases

h. REF - the noun to which a pronoun refers

Final adjustments are made to the record structure, erasing some unwanted attributes, and generally cleaning up the record structure before it moves on to the next component of the system. The record structure which is output can be used by machine translation devices to provide more accurate translations of one natural language into another. The output can also be applied to advanced text critiquing, improved information retrieval, electronic mail routing and filing, automatic text generation, and any other NLP applications that require a basic semantic analysis.

<u>Brief Description of the Drawings</u>

The foregoing and other advantages of the invention will be more fully appreciated with reference to the accompanying figures.

Figs. 1A and 1B illustrate prior art methods for the determination of missing arguments, according to current linguistic theories.

Fig. 2 depicts the relationship of Figs. 2A-K illustrating the flow diagram of the method for determination of argument structures.

Fig. 2A shows the portion of the flow diagram which processes infinitival and participial complements.

Fig. 2B is the portion of the flow diagram which processes long-distance arguments.

Fig. 2C is the portion of the flow diagram which processes passive constructions.

Fig. 2D is the portion of the flow diagram which processes indirect object constructions where the indirect object is in a prepositional phrase with the preposition "to."

Fig. 2E is the portion of the flow diagram which links predicate adjectives with their subject noun phrases.

Fig. 2F is the portion of the flow diagram which processes verb-phrase modifiers.

Fig. 2G is the portion of the flow diagram which processes noun-phrase modifiers.

Fig. 2H is the portion of the flow diagram which processes the modifiers of adjective phrases.

Fig. 2I is the portion of the flow diagram which processes the modifiers of adverb phrases.

Fig. 2J is the portion of the flow diagram which processes coordinated verb phrases.

Fig. 2K is the portion of the flow diagram which processes coordinated noun phrases.

Fig. 3 shows the parse tree and a display of the completed basic semantic structure (argument structure) for the sentence "Alice, Peter said that John kissed," illustrating the proper resolution of a long-distance dependency.

Fig. 2 shows an overall configuration of the present system. It comprises an input stage; an analysis stage that produces the syntactic structure; a preliminary post-analysis step in which arguments are assigned in the cases when semantic arguments are identical to syntactic arguments (1); the main post-processor (2), which determines the arguments in all other cases, and which itself comprises a step of assigning semantic arguments (2.1) and another step of assigning adjuncts (2.2); and finally the completed argument structure (3), which is the basic semantic structure corresponding to the input string.

During the input stage, a string of words is entered into the system. This is usually done by typing the text at a keyboard, although the string may be entered by any other means. For example, if a speech recognition component were available, the text could be entered by human voice.

The syntactic analysis stage is accomplished by the syntactic parsing component, PEG. PEG produces, for each input string, a syntactic description in the form of an attribute-value "record" structure. A more readable syntactic parse "tree" is also displayed from the information contained in the record structure. PEG has access to a very large English vocabulary list (often called a lexicon), and to the standard morphological rules of English. However, the lexicon, in combination with these rules, provides only limited morphological and syntactic information to PEG, of the following sort:

* the orthographic form (spelling) of most English words;
* parts of speech that each word may have in English;
* information about tense, number, and so forth -- that is, morphological information -- for each part of speech given for each word;
* information about the various syntactic subcategorization classes that each word might belong to (for example, whether a particular verb can be transitive or not).

PEG is a large program written in PLNLP (the Programming Language for Natural Language Processing). It consists of about 200 augmented phrase structure rules driven by a bottom-up, fully parallel processing algorithm. These rules produce the syntactic analysis of the input string. Whether a particular rule will be applicable to a particular string, or part of that string, is governed by the presence or absence of certain attributes, and their values, in the record structure covering that string. Some attributes are provided by the lexicon, and some are added by the rules themselves. What PEG does is to produce a syntactic description of a string by starting with the records for individual words, and incrementally building a larger and larger record structure, until finally a structure is arrived at which is the analysis of the entire input string. It is noteworthy that PEG uses only morphological and syntactic information, and no semantic information at all (see above), to make the analysis.

A lot of information is contained in the attribute-value analysis structures. Some of this information is simple: for example, a PAST attribute in the record for a verb phrase may have a value of "on" or "off." If it is "on," it indicates that that verb phrase is in the past tense. Some information is more complicated: for example, a verb phrase may have an attribute SUBJECT, which has as its value a pointer to another entire record structure that covers the noun phrase acting as its subject, with all of the information pertaining to that noun phrase.

All attributes assigned by PEG are syntactic attributes. Among these are some (like SUBJECT) that refer to the syntactic arguments of the input string. The first stage of semantic post-processing, according to the method disclosed herein, is to identify semantic arguments when these correspond exactly to the syntactic arguments. To do this, the post-processor simply adds a semantic attribute to the record and makes it point to the same value as its syntactic counterpart. (DSUBJECT points to the same record that SUBJECT points to, for example.)

The next stage of post-processing handles all the remaining cases of argument identification.

First, the post-processor scans the record structure to look for infinitival or participial verb complements, or for participial subject complements that appear at the end of a sentence, rather than adjacent to their subject noun phrases. Suppose it finds a present participle that is a verb complement (4). For example, in the sentence

John, in my opinion, likes entertaining women.

the words "entertaining women" form a present participle clause, a complement of the main verb "likes." The semantic object (DOBJECT) of that clause is the noun "women"; "women" is present as a syntactic object, and has been identified as the semantic object by the first stage of post-processing. But there is no syntactic subject for "entertaining women." Furthermore, it is not possible, at this stage, to decide on the correct semantic subject for "entertaining women," because we have no idea who is doing the entertaining. It might be

John who is entertaining the women, or it might be someone entirely different. There is even the possibility that the women might be entertaining John, in which case this would not be a participle clause, but a noun phrase with "entertaining" serving as adjective. Therefore a dummy semantic subject ('XX') is assigned as the value of the attribute DSUBJECT in this case. This completes the argument assignment for the verb "entertain" in the sentence above. In some later component of the system, the most likely DSUBJECT for "entertain" will be computed.

For all other cases in Fig. 2A, if a DSUBJECT already exists in the clause, then the procedure stops; otherwise, the syntactic subject of the parent clause is assigned as the DSUBJECT of the clause in question (5). As an example of a participial subject complement, consider the sentence

Mary, as you predicted, arrived excitedly waving her hands.
"Waving her hands" is not a complement of the main verb "arrived." It is a participle clause that goes with, or complements, the subject of the main clause, "Mary." "Waving" has a syntactic object ("her hands"), which is immediately identified as its semantic object, or DOBJECT. However, there is no syntactic subject for "waving her hands." Step (5) identifies "Mary" as its semantic subject (DSUBJECT).

Fig. 2B presents a flow diagram for the procedure that handles long-distance arguments. Currently four different long-distance situations are treated:

a. topicalizations: "Alice, Peter said that John kissed."

b. wh-questions: "Who did Peter say that John kissed?"

c. relative clauses: "This is the girl who Peter said that John kissed."

d. free relatives: "I know who Peter said that John kissed.

The procedure identifies the fronted long-distance element (e.g., the italicized words in the sentences above), and calls this element FRNTNP. Then it identifies the clause that is the candidate for the long-distance relationship, and calls this CAND. CAND is usually the final clause in a list of clauses post-modifying the main verb, said list including complement clauses and infinitive clauses, but not, for example, subordinate clauses introduced by a subordinate conjunction.

If CAND itself contains coordinated verb phrases (6), as in

Who did Peter say that John kissed and hugged? then it is sent to the procedure that separates coordinated verb phrases (see Fig. 2J). Eventually the separated elements will be routed back to the long-distance procedure. The next consideration is whether or not CAND contains a dangling preposition (7), as in

Who did Peter say that John ran into?

In the foregoing sentence, CAND is "John ran into" and the dangling preposition is "into." In a case like this one, FRNTNP will be assigned as the object of the dangling preposition ("John ran into who?"), and the procedure terminates. The clause is available for further argument identification, if necessary.

If there is no coordination in CAND, and no dangling preposition in CAND, then the main business of this procedure begins.

A complex set of information is necessary to identify properly the argument for a long-distance dependency (8). First, the subcategorization class of the main verb in CAND is important: is it complex-transitive, ditransitive, or neither? A complex-transitive verb takes an object and an object complement: "They elected him (object) President (object complement)." A ditransitive verb takes an indirect object and an object: "They gave her (indirect object) an award (object)." Second, is CAND a tensed clause or an infinitive clause? Third, how many arguments are already present in the clause? Fourth, what features are present on the arguments that are already present, as well as on the long-distance element (FRNTNP)?

The feature that is of particular interest here, HUM, is attached to some pronouns ("who," "I," "you," "he," "she," etc.) and to some nouns ("man," "woman," "child," etc.). Although the feature clearly has a meaningful interpretation, it should not be regarded, for present purposes, as semantic. It is simply a feature, like any other feature, that can be attached to members of a list of words. It indicates a high probability that the word to which it is attached will behave, syntactically, in a certain way under certain circumstances. Therefore the existence of this feature, which comes from the lexicon, is not a contradiction to the earlier claim that the syntactic parsing component works without semantic information.

Based on the information that has been described, the procedure resolves the long-distance dependency (9). In the sentence

Who did John want to kiss?
FRNTNP ("who") is assigned as the DOBJECT of the verb "kiss"; it is most directly affected by the verb's action. In the sentence
Who did John want to write?
FRNTNP ("who") is assigned as the DINDOBJ of the verb "write"; it receives something through the verb's action. In the sentence
What did John want to write?
FRNTNP ("what") is assigned as the DOBJECT of the verb "write."

When a passive construction is encountered

(Fig. 2C), the post-processor again considers whether or not there is a dangling preposition in the passivized clause (10), for example:
The house was broken into.

If a dangling preposition is present, like "into" in the foregoing sentence, the syntactic subject ("the house" in this case) will be assigned as the object of the dangling preposition ("broken into the house"). Then the procedure skips ahead to locate a possible "by"-prepositional phrase (13), and to identify a semantic subject.

If there is no dangling preposition, a complex set of information is consulted to identify properly the semantic arguments for a passive construction (11). First, the subcategorization class of the main verb is important: is it complex-transitive, ditransitive, or neither? (See above.) Second, how many arguments are already present in the clause? Third, what features are present on the syntactic subject of the clause?

The feature that is of particular interest here, ANIM, is very like HUM in all respects except that it attaches to nouns that most probably refer to living beings, not just human beings. Again, although the feature clearly has a meaningful interpretation, it should not be regarded as semantic in its present use.

Next, the procedure assigns semantic arguments other than DSUBJECT, based on the information that has been described (12). In the passive sentence
The man was elected President.
"the man" is the DOBJECT, and "President" is the DOBJCOMP. In the sentence
He was given a mandate.
"he" is the DINDOBJ (receiving something), and "a mandate" is the DOBJECT (the thing received). In the sentence
You were invited.
"you" is the DOBJECT; there is no DOBJCOMP or DINDOBJ.

The procedure moves on to assign a DSUBJECT for the passive construction. It checks to see if a "by"-prepositional phrase exists (13). If not, the DSUBJECT is assigned a value of 'XX', which can be resolved later, if and when the necessary information is processed. If the "by"-PP exists, then DSUBJECT is assigned to point to the object of that prepositional phrase (14). For example, in the sentence
You were invited by the President.

"the President" is the DSUBJECT. This sentence is the passive equivalent of the active sentence "The President invited you." In both sentences, the DSUBJECT is "the President" and the DOBJECT is "you." In this manner, although the surface syntactic forms of active and passive sentences are quite different, the underlying argument structure demonstrates their semantic similarity.

The next sub-procedure (Fig. 2D) handles the indirect object construction. The purpose of this procedure is to demonstrate the semantic similarity between sentences like "We gave him a mandate" and "We gave a mandate to him." In this respect, the indirect object procedure has the same purpose as the passive procedure. It is, however, much simpler. The first step is to locate the object of the "to"-prepositional phrase; the next step is to assign this object as the value of the DINDOBJ attribute. When this step has been completed, both of the sentences under discussion will have a DSUBJECT of "we," a DINDOBJ of "him," and a DOBJECT of "a mandate."

This concludes the first step (2.1) of the central part of the post-processor, in which semantic attributes are assigned their values, and an argument structure is built for the input string. The second step (2.2) involves the assignment of semantic adjuncts -- modifiers that are not arguments, but that are nevertheless important for the final semantic structure.

The first phase in this second step is to link predicate adjectives with their subject noun phrases (Fig. 2E). An example of a predicate adjective construction is the sentence

Mary and John are, and always will be, happy.
The adjective "happy" applies to both "Mary" and "John" in the subject noun phrase "Mary and John"; but it is quite far removed from that NP. If the phrase were "happy John and happy Mary," there would be no problem. But the syntactic analysis of the foregoing sentence does not make the proper connection directly, because of the displacement of the adjective from the subject. In this respect, the problem of linking predicate adjectives with their NPs is like the problem of long-distance dependencies.

The first step in solving this problem is to construct a list of the subject noun phrases (15). If there is only one subject NP, then the list will be a list with one member. On each member of the list, a PADJ attribute is created, with its value a pointer to the predicate adjective (16). Then the next member of the list of subject NPs is processed. When the list is empty, the procedure terminates.

All arguments having been identified, the post-processor considers the major phrase categories VP, NP (including PP), AJP, and AVP, to make sure that all non-argument modifiers are assigned to their proper semantic attributes. (These non-argument modifiers are also called "adjuncts.")

The first category to be considered is the verb phrase (Fig. 2F). If a verb phrase itself contains coordinated VPs, then it is sent to the procedure that separates coordinated VPs (17). The separated elements will be routed back to the main procedure. If the VP is not a coordinated segment, then a list of all modifiers, both pre-modifiers and post-modifiers, is constructed (18). For each member of that list,

a. If it is a NP, then it is sent to the procedure that handles NP modifiers (19).

b. If it is a PP, then

1. An attribute is created on the VP, this attribute having the same name as the preposition in the PP (20);

2. The object of the PP is assigned as the value of this attribute (20);

3. The segment is sent to the procedure that handles NP modifiers (see Fig. 2G).

c. If it is an adjective phrase or an adverb phrase, then a MODS attribute is created on the VP, with its value being a pointer to the AJP or AVP (21).

d. If it is an embedded clause (for example, a subordinate clause), then a PROP attribute is created on the VP, with its value being a pointer to the embedded clause (22).

Then the next member of the list of modifiers is processed (23). When the list is empty, the procedure terminates.

The next category to be considered is the noun phrase (Fig. 2G). If a noun phrase itself contains coordinated NPs, then it is sent to the procedure that separates coordinated NPs (24). The separated elements will be routed back to the main procedure. If the NP is not a coordinated segment, then a list of all modifiers, both pre-modifiers and post-modifiers, is constructed (25). For each member of that list,

a. If it is a determiner or quantifier (words like "the," "a," "this," "some," "all," etc.), then an OPS ("operators") attribute is created on the NP, with its value being a pointer to the determiner or quantifier (26).

b. If it is an adjective phrase other than a determiner or quantifier, then a NADJ attribute is created on the NP, with its value being a pointer to the adjective phrase (27).

c. If the NP is a gerund (noun ending in "-ing"), then,

1. If the phrase being considered is a possessive adjective, then a DSUBJECT attribute is created on the NP, with its value being a pointer to the possessive adjective (28).

2. If there is no possessive adjective in the gerundive NP, then a DSUBJECT attribute is created on the NP, with its value being 'XX' (29). This value can be changed when more information is available.

d. If the phrase being considered is an embedded clause (for example, a relative clause), then a PROP attribute is created on the VP, with its value being a pointer to the embedded clause (30).

e. If it is a PP, then

1. An attribute is created on the NP, this attribute having the same name as the preposition in the PP (31);

2. The object of the PP is assigned as the value of this attribute (31);

3. The segment is sent back to the procedure that handles NP modifiers (that is, to the procedure in Fig. 2G, currently being discussed).

f. If the phrase being considered is something other than those mentioned here, then a MODS attribute is created on the NP, with its value being a pointer to this other phrase (32).

Then the next member of the list of modifiers is processed (33). When the list is empty, the procedure terminates.

For identifying the semantic modifiers on adjective phrases and adverb phrases (Figs. 2H and 2I), the procedure steps are identical. First a list of all modifiers is constructed (34). For each member of that list,

a. If it is a PP, then

1. An attribute is created on the AJP or AVP, this attribute having the same name as the preposition in the PP (35);

2. The object of the PP is assigned as the value of this attribute (35);

3. The segment is sent to the procedure that handles NP modifiers (see Fig. 2G).

b. Otherwise, a MODS attribute is created on the AJP or AVP, with its value being a pointer to the list member under consideration (36).

Then the next member of the list of modifiers is processed (37). When the list is empty, the procedure terminates.

Fig. 2J illustrates the process of separating coordinated verb phrases. This procedure is called from the procedure that handles VP modifiers (see Fig. 2F). First a list is constructed of all VPs that are within the larger coordinated VP (38). Then, for each member of that list of VPs,

a. If the coordinated VPs have a common syntactic subject, then that subject is distributed to each member VP in the list (39). An example is John came in and sat down.
In the foregoing sentence, "John" is the syntactic subject for both of the coordinated VPs "came in" and "sat down."

b. If the clause is passive (e.g., "John was hugged and kissed"), then a PASSIVE attribute is set "on" in each member VP (40).

c. If there is a syntactic direct object in the final VP, then that object is distributed to each transitive member VP in the list (41). An example is John wrote and signed the document.
In the above sentence, "the document" is the

syntactic object for both of the coordinated VPs "wrote" and "signed."

d. If there is a long-distance dependency in the clause, then the fronted element is distributed to each member VP in the list (42). An example is What did Peter say that John wrote and signed? In the above sentence, the fronted question word "what" is distributed to the VP "wrote" and to the VP "signed." The dependency will later be resolved by the procedure displayed in Fig. 2B.

e. After all of the above distributions have been made, each individual member of the VP list is sent to the main procedure for identifying arguments (43). Then the next member of the list of VPs is processed (44). When the list is empty, the procedure terminates.

Fig. 2K illustrates the process of separating coordinated noun phrases. This procedure is called from the procedure that handles NP modifiers (see Fig. 2G). First a list is constructed of all NPs that are within the larger coordinated NP (45). Then, for each member of that list of NPs,

a. A MODS attribute is created on the parent NP, with its value being a pointer to the NP member of the list (46). This results in having all coordinated NPs listed as MODS under their parent NP.

b. After each individual member of the NP list has been so assigned, it is sent to the main procedure for identifying arguments (47). Then the next member of the list of NPs is processed (48). When the list is empty, the procedure terminates.

Fig. 3 displays two structural stages that occur in the processing of a sentence with a long-distance dependency: "Alice, Peter said that John kissed." First is the syntactic structure, presented in abbreviated form as a parse tree (49). The tree is produced by using only a few of the attributes and values that actually exist in the record structure after PEG has processed this input string. In this parse tree, "Alice" is displaced from the verb "kissed"; and there is no indication of any meaningful relationship between those two words.

Second is the basic semantic structure, the argument structure, presented in abbreviated form as a kind of chart (50). This chart is produced by using only the semantic attributes that have been disclosed in this invention, and by indenting them to group the arguments properly with their verbs. In this argument structure, "Alice" is correctly linked as the DOBJECT of the verb "kissed."

The record structure which is output can be used by machine translation devices to provide more accurate translations of one natural language into another. The output can also be applied to advanced text critiquing, improved information retrieval, electronic mail routing and filing, automatic text generation, and any other NLP applications that require a basic semantic analysis.

The invention is embodied as a computer program running on a data processing system such as that disclosed in US Patent 4,731,735 to Borgendale, et al., cited above and incorporated herein by reference. The program embodying the invention is stored in the memory in the system and is executed by the execution unit. The string of natural language words can be input to the execution unit from the keyboard, from the bulk storage, from the connected terminals or from the communications link. The syntactic parsing stage and the semantic analysis stage of the invention are executed by the program embodiment of the invention in the data processing system. The semantic attribute record structure output by the program embodiment of the invention can be output to the display, to the printer, to the bulk storage, to the communications link or to another partition in the memory, as a semantic characterization of the input string which can be immediately displayed to the user on the display screen, or which can be input to utilization processes or programs running on the same data processing system or on other data processing systems.

Claims

1. A computer method for determining basic semantic structures for natural language word strings, the method comprising the steps of:
inputting a string consisting of a plurality of words forming a linguistic expression in a natural language;
parsing the input string with a syntactic set of rules to derive a syntactic structure for the string, identifying syntactic arguments for said words;
identifying a first group of words in said string as semantic arguments, when said semantic arguments correspond exactly to said syntactic arguments;
identifying semantic arguments in a second group of words in said string, which are not in said first group, by the following steps:
assigning missing arguments of infinitive clauses and participial clauses;
assigning long distance arguments;
assigning missing or displaced arguments in passive constructions;
assigning arguments for indirect object construction;
outputting said assigned arguments in a record which provides a normalized semantic structure for said input word string.

2. The computer method for determining basis

semantic structures for natural language word strings of claim 1, which further comprises the steps of:
after said step of assigning arguments for indirect object construction, the steps of:
linking predicate adjectives to their subject noun phrases;
linking verb phrase modifiers to their verbs;
linking noun phrase modifiers to their nouns;
linking adjective phrase modifiers to their adjectives;
linking adverb phrase modifiers to their adverbs;
whereby modifier words are semantically linked in said input word string.
3. A computer method for determining semantic structures according to claim 1, where, in said assigning steps, basic semantic structures are created by procedures that traverse said syntactic structure containing all of the syntactic information accumulated in said parsing step.
4. A computer method for determining semantic structures according to claim 3, wherein said normalized semantic structure is achieved by adding new attributes and values having semantic significance, to said syntactic structure.
5. A computer method for determining basic semantic structures for natural language word strings, the method comprising the steps of:
inputting a string consisting of a plurality of words forming a linguistic expression in a natural language;
parsing the input string with a syntactic set of rules to derive a syntactic structure for the string, identifying syntactic arguments for said words;
identifying semantic arguments in said string by the following steps:
assigning missing arguments of infinitive clauses and participial clauses;
assigning long distance arguments;
assigning missing or displaced arguments in passive constructions;
assigning arguments for indirect object construction;
outputting said assigned arguments in a record which provides a normalized semantic structure for said input word string.
6. The computer method for determining basis semantic structures for natural language word strings of claim 5, which further comprises the steps of:
after said step of assigning arguments for indirect object construction, the steps of:
linking predicate adjectives to their subject noun phrases;
linking verb phrase modifiers to their verbs;
linking noun phrase modifiers to their nouns;
linking adjective phrase modifiers to their adjectives;
linking adverb phrase modifiers to their adverbs;
whereby modifier words are semantically linked in said input word string.
7. A computer method for determining semantic structures according to claim 5, where, in said assigning steps, basic semantic structures are created by procedures that traverse said syntactic structure containing all of the syntactic information accumulated in said parsing step.
8. A computer method for determining semantic structures according to claim 7, wherein said normalized semantic structure is achieved by adding new attributes and values having semantic significance, to said syntactic structures.
9. A computer method for determining basic semantic structures for natural language word strings, the method comprising the steps of:
inputting a string consisting of a plurality of words forming a linguistic expression in a natural language;
parsing the input string with a set of syntactic rules which are free of semantic information, to derive a syntactic structure for the string, identifying syntactic arguments for said words;
identifying semantic arguments in said string by the following steps:
assigning missing arguments of infinitive clauses and participial clauses;
assigning long distance arguments;
assigning missing or displaced arguments in passive constructions;
assigning arguments for indirect object construction;
outputting said assigned arguments in a record which provides a normalized semantic structure for said input word string.
10. The computer method for determining basis semantic structures for natural language word strings of claim 9, which further comprises the steps of:
after said step of assigning arguments for indirect object construction, the steps of:
linking predicate adjectives to their subject noun phrases;
linking verb phrase modifiers to their verbs;
linking noun phrase modifiers to their nouns;
linking adjective phrase modifiers to their adjectives;
linking adverb phrase modifiers to their adverbs;
whereby modifier words are semantically linked in said input word string.
11. A computer method for determining semantic structures according to claim 9, where, in said assigning steps, basic semantic structures are created by procedures that traverse said syntactic structure containing all of the syntactic information accumulated in said parsing step.
12. A computer method for determining semantic

structures according to claim 11, wherein said normalized semantic structure is achieved by adding new attributes and values having semantic significance, to said syntactic structures.

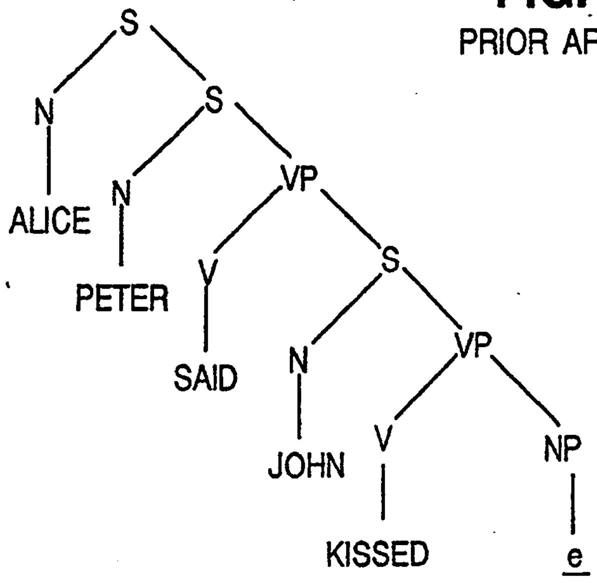


FIG. 1A
PRIOR ART (EC)

FIG. 1B
PRIOR ART (FU)

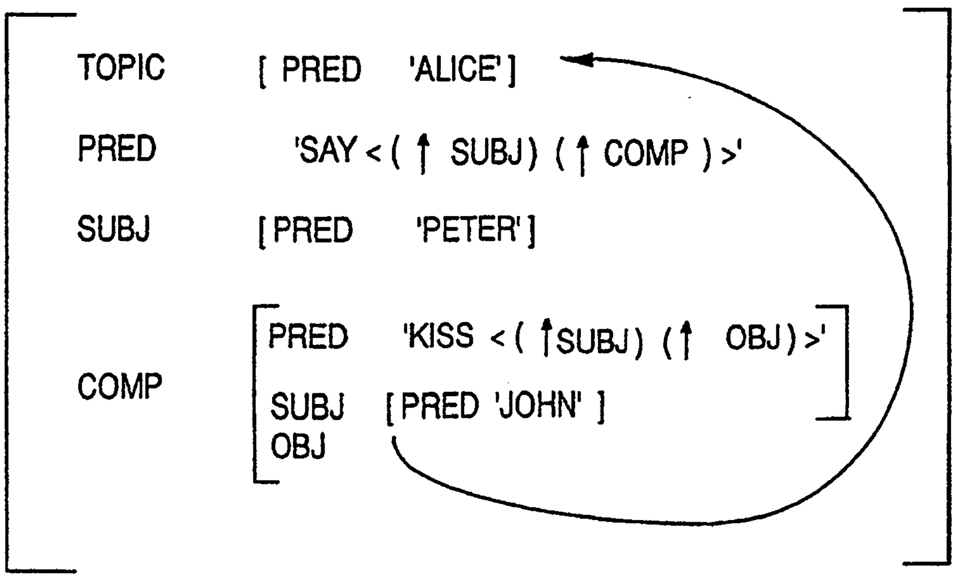
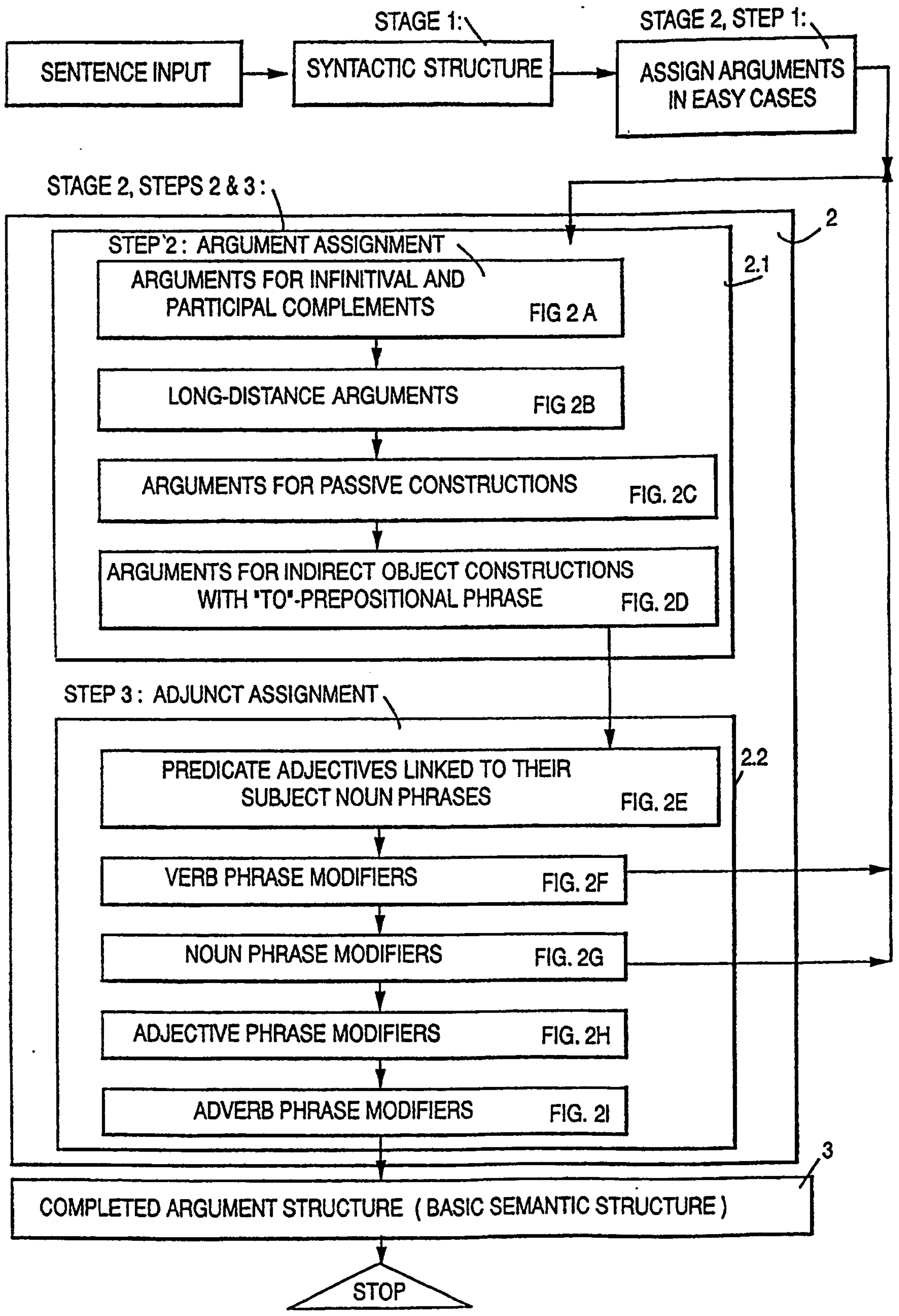

TOPIC [PRED 'ALICE']
PRED 'SAY < (↑ SUBJ) (↑ COMP) >'
SUBJ [PRED 'PETER']
COMP
PRED 'KISS < (↑SUBJ) (↑ OBJ) >'
SUBJ [PRED 'JOHN']
OBJ

FIG. 2



STAGE 1:
STAGE 2, STEP 1:
SENTENCE INPUT
SYNTACTIC STRUCTURE
ASSIGN ARGUMENTS IN EASY CASES
STAGE 2, STEPS 2 & 3 :
2
STEP 2 : ARGUMENT ASSIGNMENT
2.1
ARGUMENTS FOR INFINITIVAL AND PARTICIPAL COMPLEMENTS FIG 2 A
LONG-DISTANCE ARGUMENTS FIG 2B
ARGUMENTS FOR PASSIVE CONSTRUCTIONS FIG. 2C
ARGUMENTS FOR INDIRECT OBJECT CONSTRUCTIONS WITH "TO"-PREPOSITIONAL PHRASE FIG. 2D
STEP 3 : ADJUNCT ASSIGNMENT
2.2
PREDICATE ADJECTIVES LINKED TO THEIR SUBJECT NOUN PHRASES FIG. 2E
VERB PHRASE MODIFIERS FIG. 2F
NOUN PHRASE MODIFIERS FIG. 2G
ADJECTIVE PHRASE MODIFIERS FIG. 2H
ADVERB PHRASE MODIFIERS FIG. 2I
3
COMPLETED ARGUMENT STRUCTURE (BASIC SEMANTIC STRUCTURE)
STOP

FIG. 2A

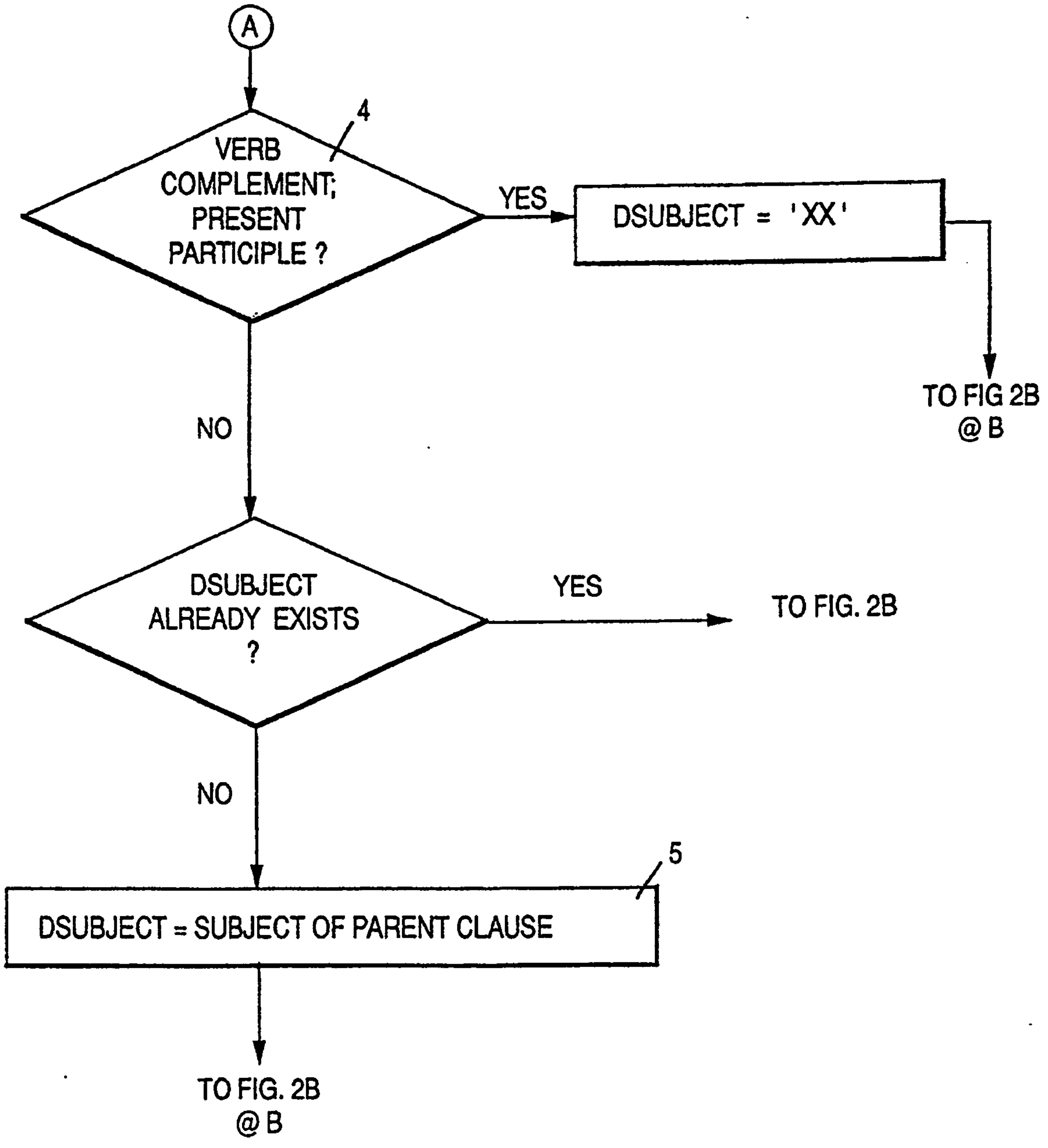

A
4
VERB COMPLEMENT; PRESENT PARTICIPLE ?
YES
DSUBJECT = 'XX'
TO FIG 2B @ B
NO
DSUBJECT ALREADY EXISTS ?
YES
TO FIG. 2B
NO
5
DSUBJECT = SUBJECT OF PARENT CLAUSE
TO FIG. 2B @ B

FIG. 2B

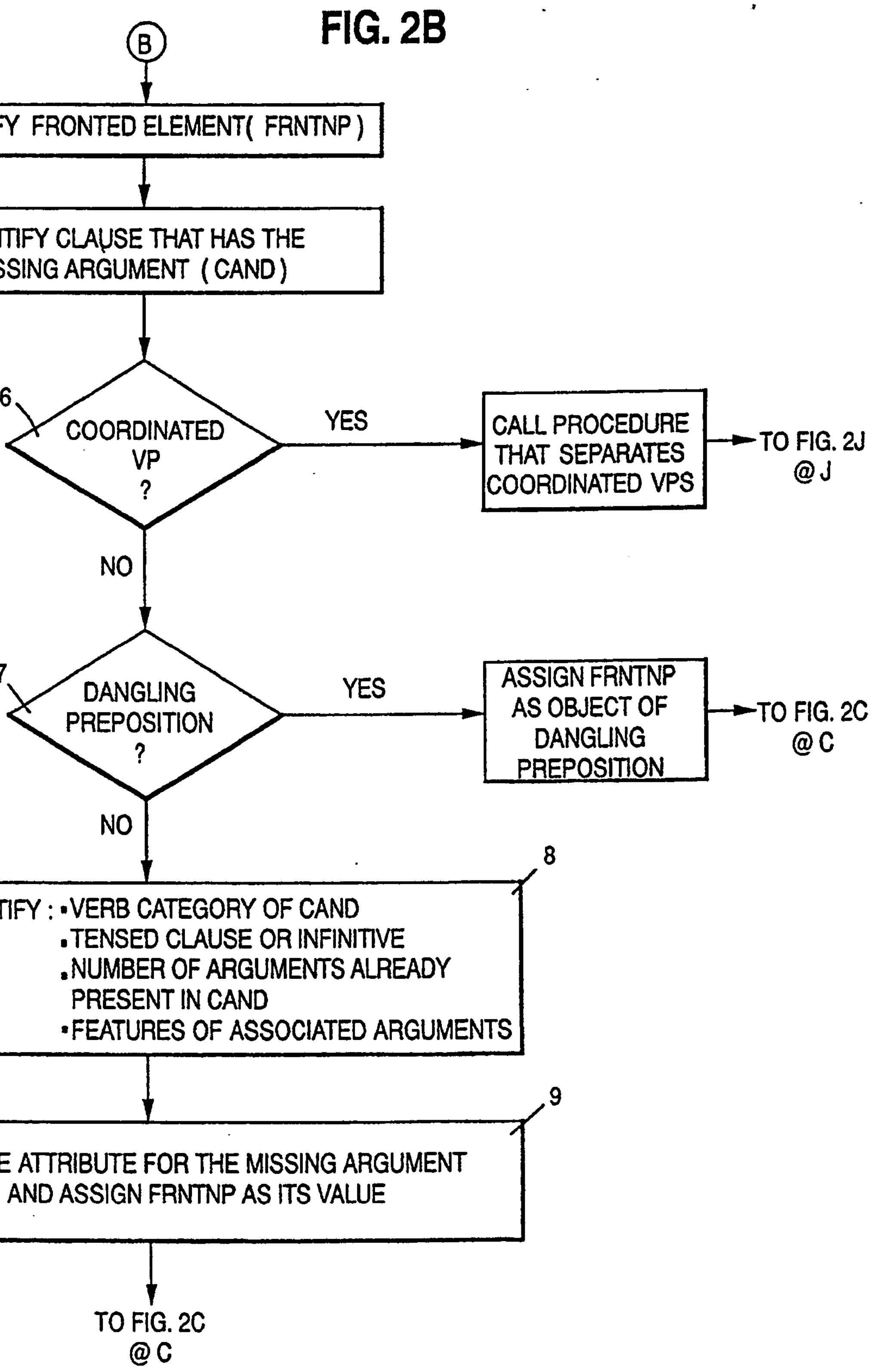

B
IDENTIFY FRONTED ELEMENT(FRNTNP)
IDENTIFY CLAUSE THAT HAS THE MISSING ARGUMENT (CAND)
6
COORDINATED VP ?
YES
CALL PROCEDURE THAT SEPARATES COORDINATED VPS
TO FIG. 2J @ J
NO
7
DANGLING PREPOSITION ?
YES
ASSIGN FRNTNP AS OBJECT OF DANGLING PREPOSITION
TO FIG. 2C @ C
NO
8
IDENTIFY : •VERB CATEGORY OF CAND
•TENSED CLAUSE OR INFINITIVE
•NUMBER OF ARGUMENTS ALREADY PRESENT IN CAND
•FEATURES OF ASSOCIATED ARGUMENTS
9
CREATE ATTRIBUTE FOR THE MISSING ARGUMENT AND ASSIGN FRNTNP AS ITS VALUE
TO FIG. 2C @ C

FIG. 2C

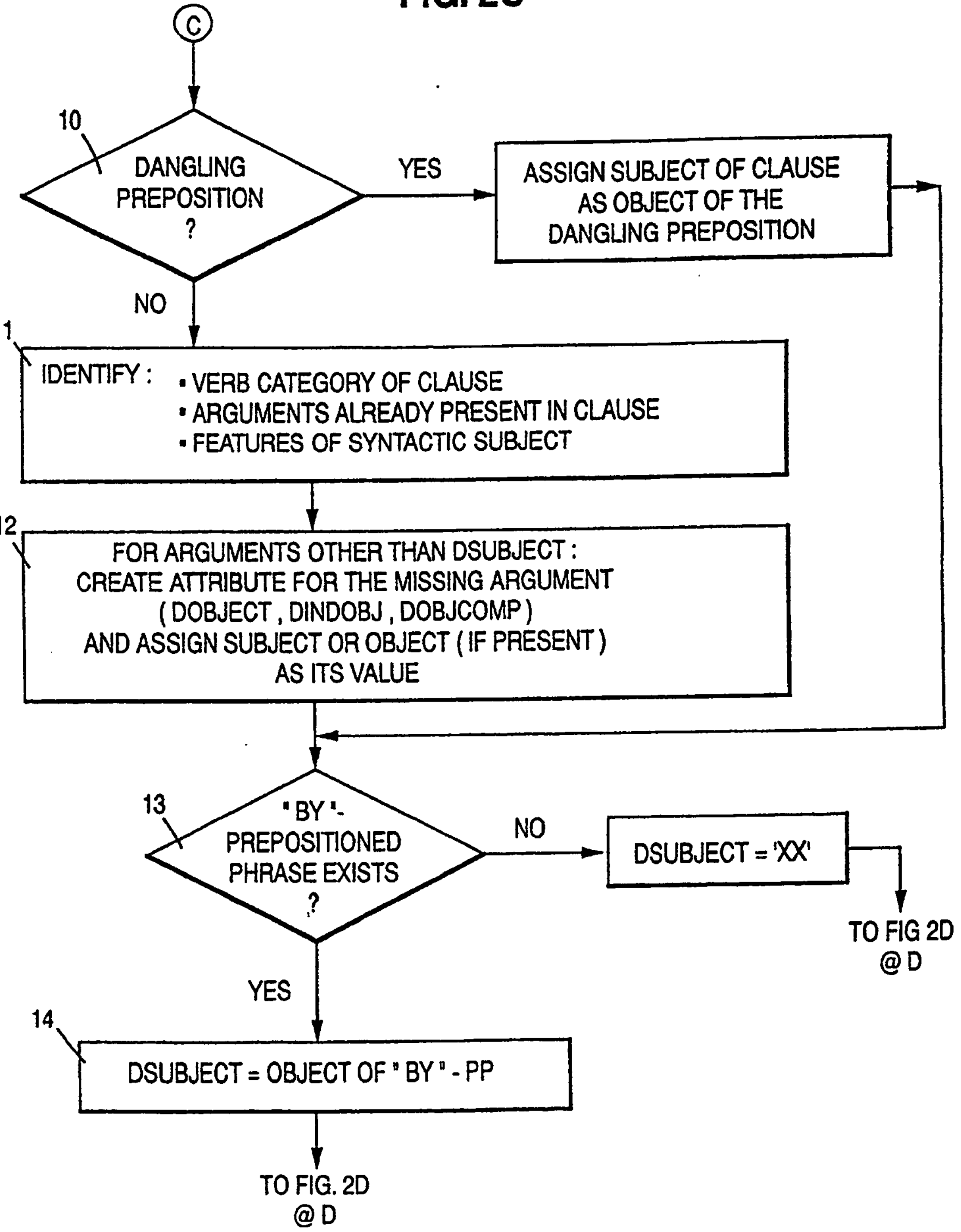

C
10
DANGLING PREPOSITION ?
YES
ASSIGN SUBJECT OF CLAUSE AS OBJECT OF THE DANGLING PREPOSITION
NO
11
IDENTIFY : • VERB CATEGORY OF CLAUSE • ARGUMENTS ALREADY PRESENT IN CLAUSE • FEATURES OF SYNTACTIC SUBJECT
12
FOR ARGUMENTS OTHER THAN DSUBJECT : CREATE ATTRIBUTE FOR THE MISSING ARGUMENT (DOBJECT , DINDOBJ , DOBJCOMP) AND ASSIGN SUBJECT OR OBJECT (IF PRESENT) AS ITS VALUE
13
" BY "- PREPOSITIONED PHRASE EXISTS ?
NO
DSUBJECT = 'XX'
TO FIG 2D @ D
YES
14
DSUBJECT = OBJECT OF " BY " - PP
TO FIG. 2D @ D

FIG. 2D

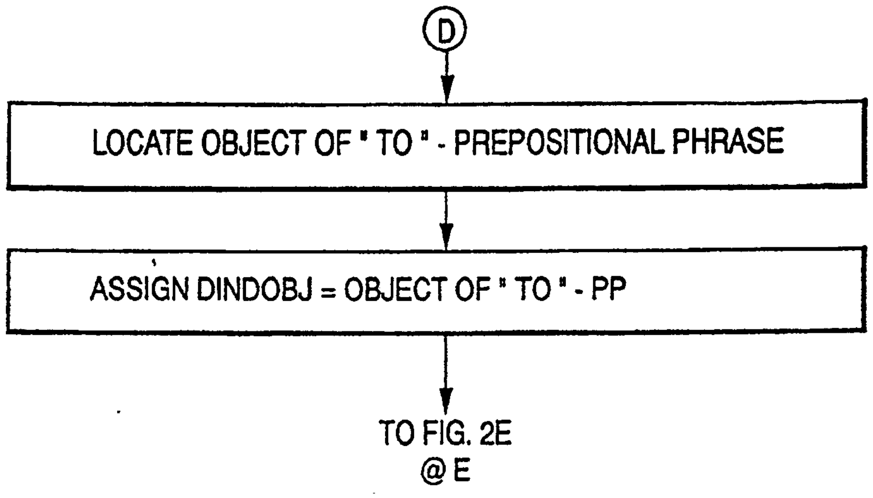

D
LOCATE OBJECT OF " TO " - PREPOSITIONAL PHRASE
ASSIGN DINDOBJ = OBJECT OF " TO " - PP
TO FIG. 2E
@ E

FIG. 2E

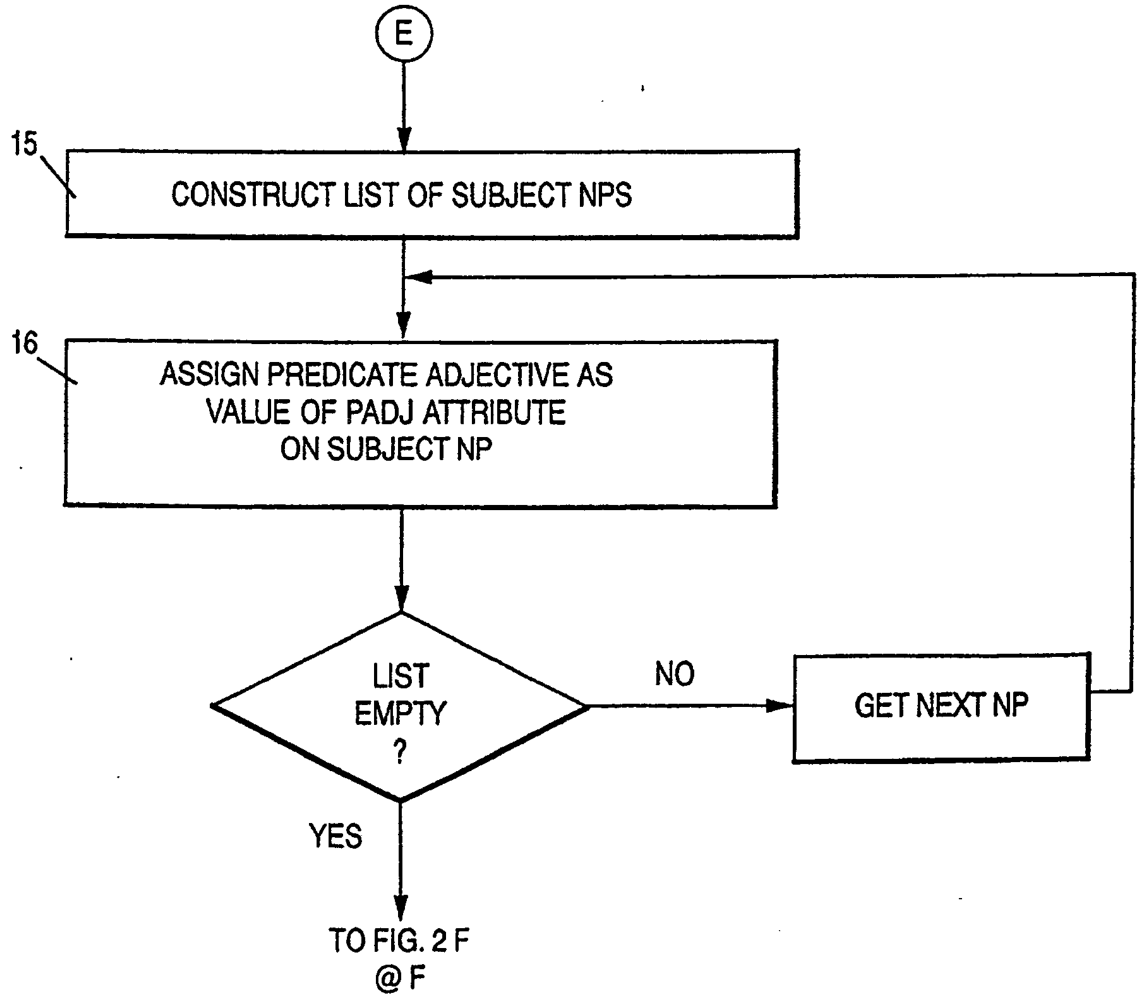

E
15
CONSTRUCT LIST OF SUBJECT NPS
16
ASSIGN PREDICATE ADJECTIVE AS VALUE OF PADJ ATTRIBUTE ON SUBJECT NP
LIST EMPTY ?
NO
GET NEXT NP
YES
TO FIG. 2 F
@ F

FIG. 2F

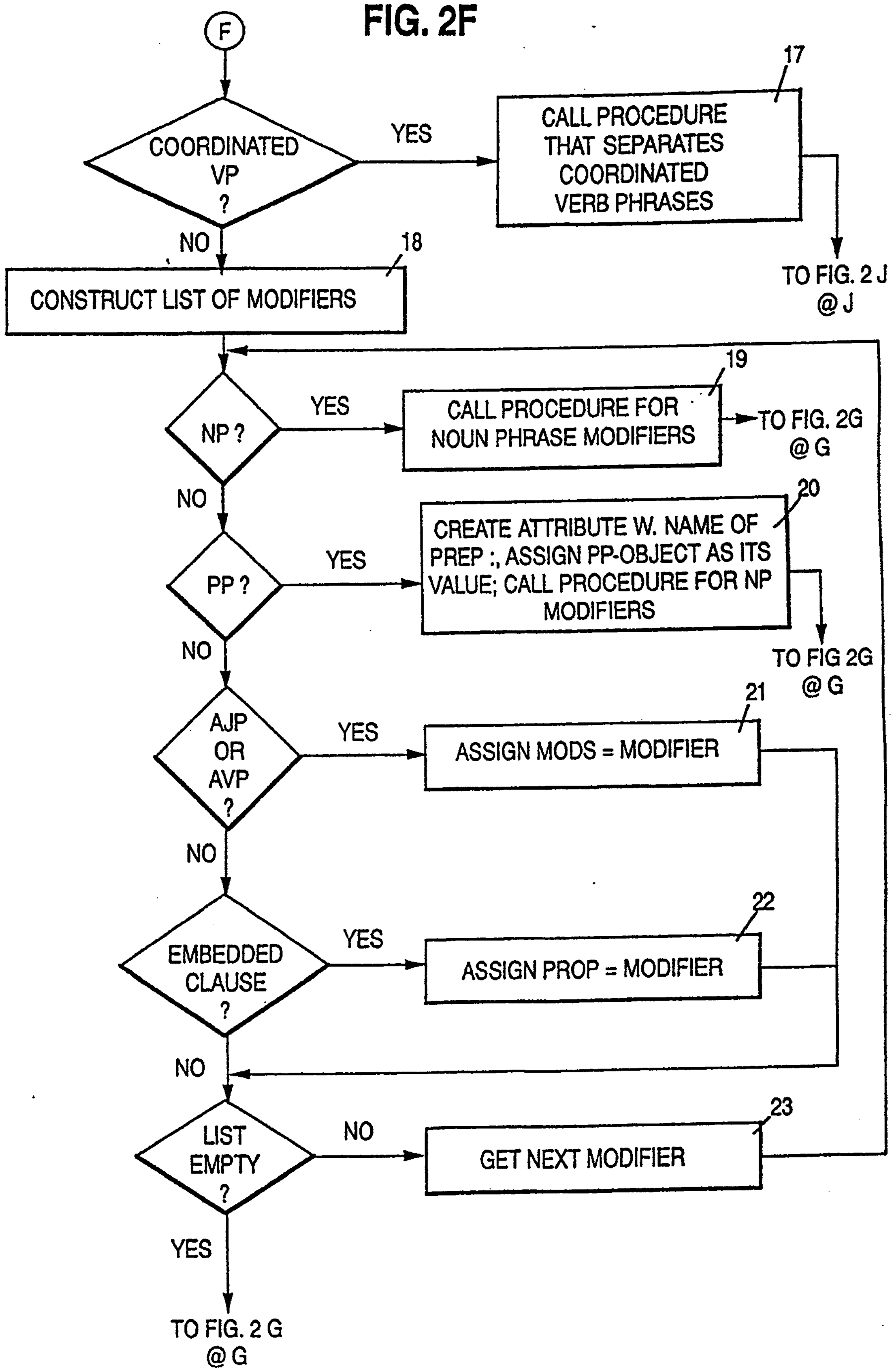

F
COORDINATED VP ?
YES
CALL PROCEDURE THAT SEPARATES COORDINATED VERB PHRASES
17
TO FIG. 2 J @ J
NO
18
CONSTRUCT LIST OF MODIFIERS
NP ?
YES
CALL PROCEDURE FOR NOUN PHRASE MODIFIERS
19
TO FIG. 2G @ G
NO
PP ?
YES
CREATE ATTRIBUTE W. NAME OF PREP :, ASSIGN PP-OBJECT AS ITS VALUE; CALL PROCEDURE FOR NP MODIFIERS
20
TO FIG 2G @ G
NO
AJP OR AVP ?
YES
ASSIGN MODS = MODIFIER
21
NO
EMBEDDED CLAUSE ?
YES
ASSIGN PROP = MODIFIER
22
NO
LIST EMPTY ?
NO
GET NEXT MODIFIER
23
YES
TO FIG. 2 G @ G

FIG. 2G

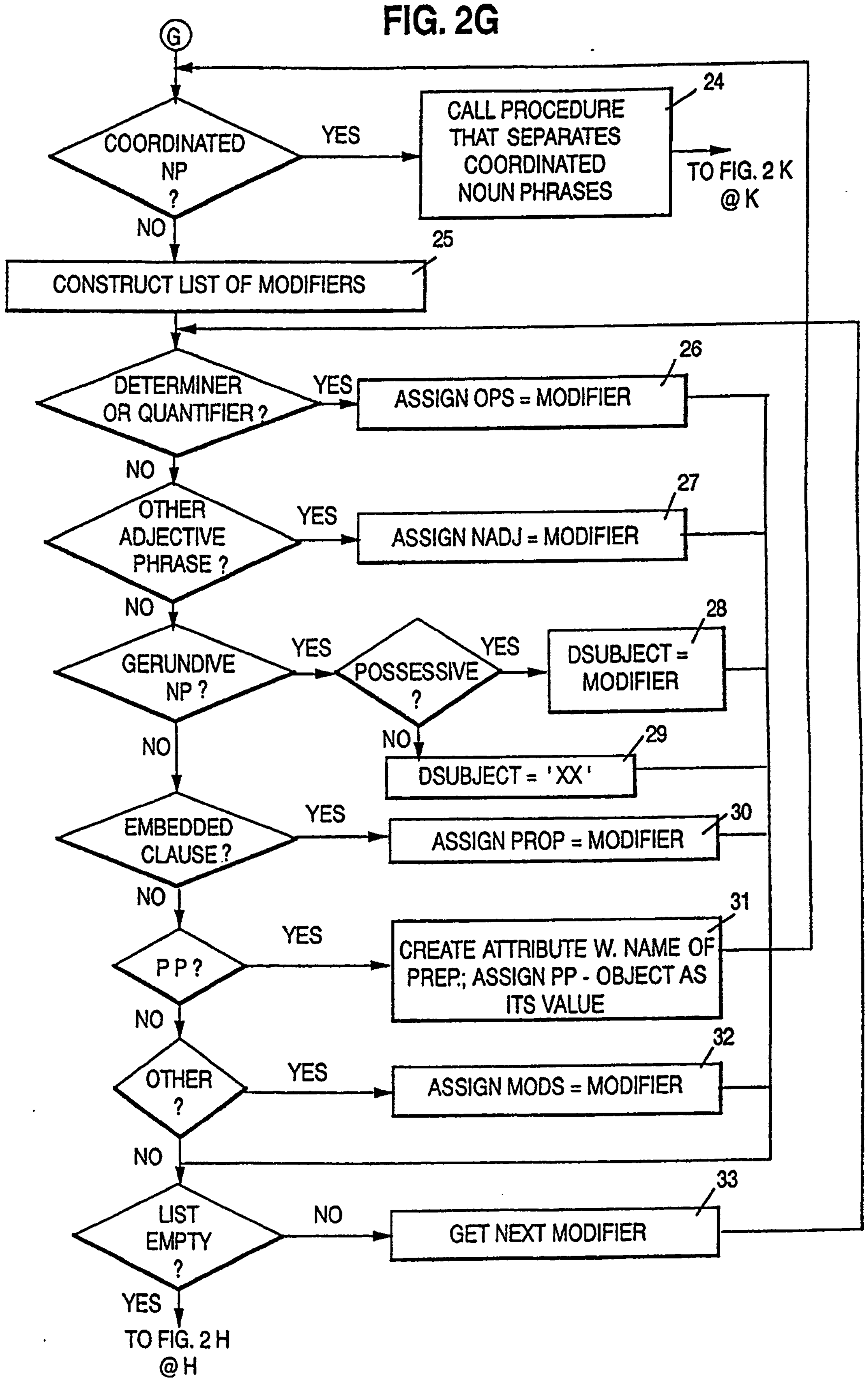

G
COORDINATED NP ?
YES
NO
CALL PROCEDURE THAT SEPARATES COORDINATED NOUN PHRASES
24
TO FIG. 2 K @ K
CONSTRUCT LIST OF MODIFIERS
25
DETERMINER OR QUANTIFIER ?
YES
NO
ASSIGN OPS = MODIFIER
26
OTHER ADJECTIVE PHRASE ?
YES
NO
ASSIGN NADJ = MODIFIER
27
GERUNDIVE NP ?
YES
NO
POSSESSIVE ?
YES
NO
DSUBJECT = MODIFIER
28
DSUBJECT = 'XX'
29
EMBEDDED CLAUSE ?
YES
NO
ASSIGN PROP = MODIFIER
30
P P ?
YES
NO
CREATE ATTRIBUTE W. NAME OF PREP.; ASSIGN PP - OBJECT AS ITS VALUE
31
OTHER ?
YES
NO
ASSIGN MODS = MODIFIER
32
LIST EMPTY ?
NO
YES
GET NEXT MODIFIER
33
TO FIG. 2 H @ H

FIG. 2H

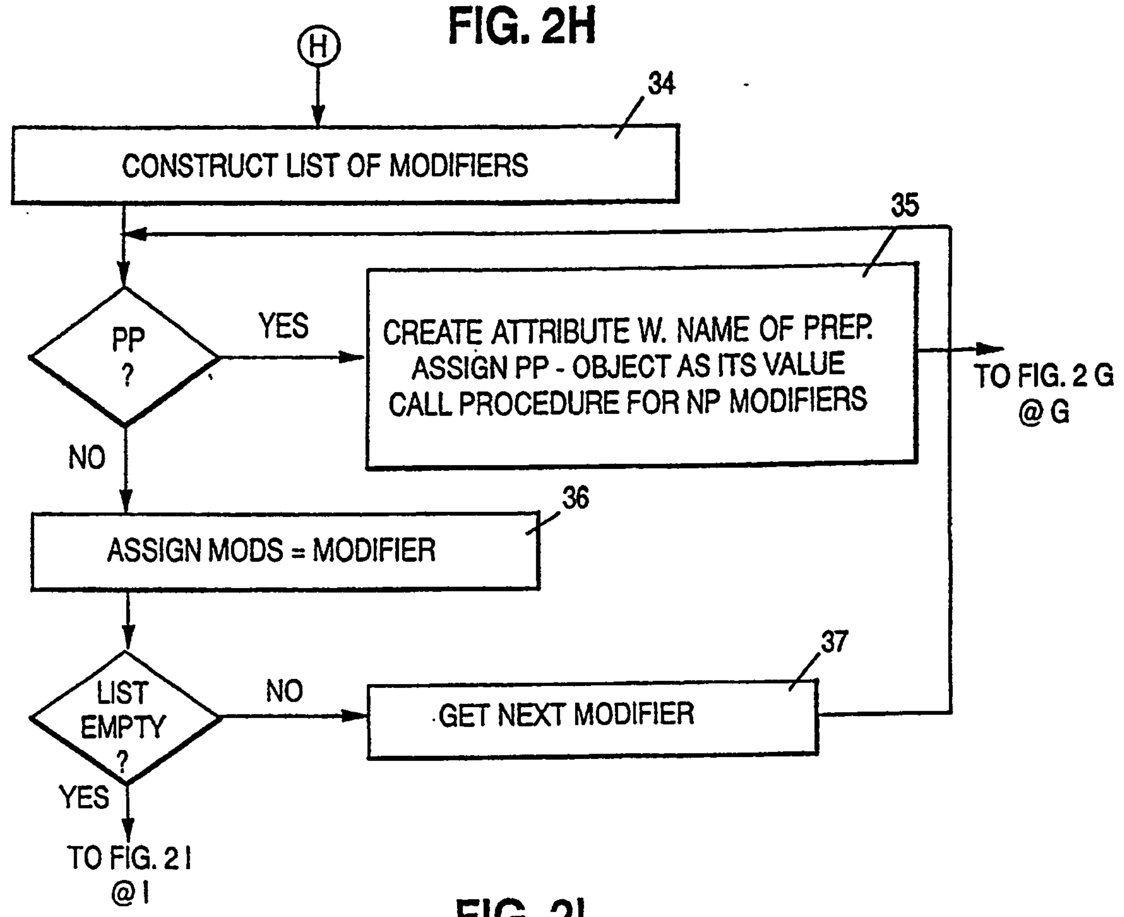

H
34
CONSTRUCT LIST OF MODIFIERS
35
PP ?
YES
CREATE ATTRIBUTE W. NAME OF PREP. ASSIGN PP - OBJECT AS ITS VALUE CALL PROCEDURE FOR NP MODIFIERS
TO FIG. 2 G @ G
NO
36
ASSIGN MODS = MODIFIER
37
LIST EMPTY ?
NO
GET NEXT MODIFIER
YES
TO FIG. 2 I @ I

FIG. 2I

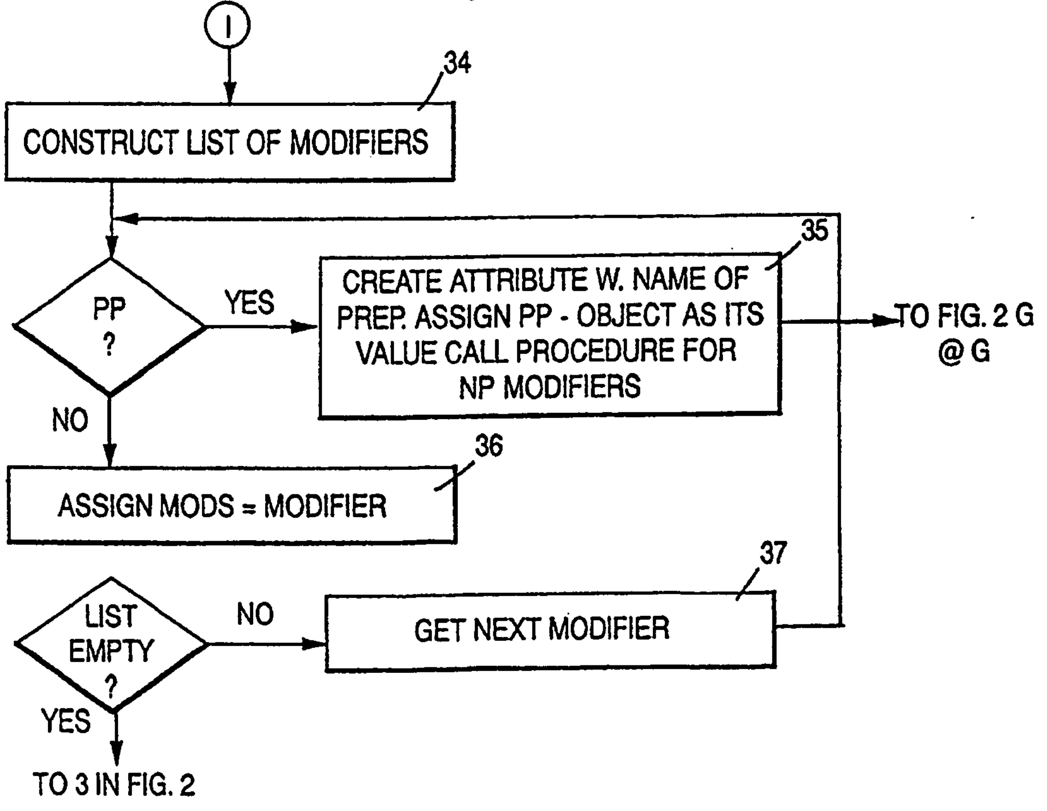

I
34
CONSTRUCT LIST OF MODIFIERS
35
PP ?
YES
CREATE ATTRIBUTE W. NAME OF PREP. ASSIGN PP - OBJECT AS ITS VALUE CALL PROCEDURE FOR NP MODIFIERS
TO FIG. 2 G @ G
NO
36
ASSIGN MODS = MODIFIER
37
LIST EMPTY ?
NO
GET NEXT MODIFIER
YES
TO 3 IN FIG. 2

FIG. 2J

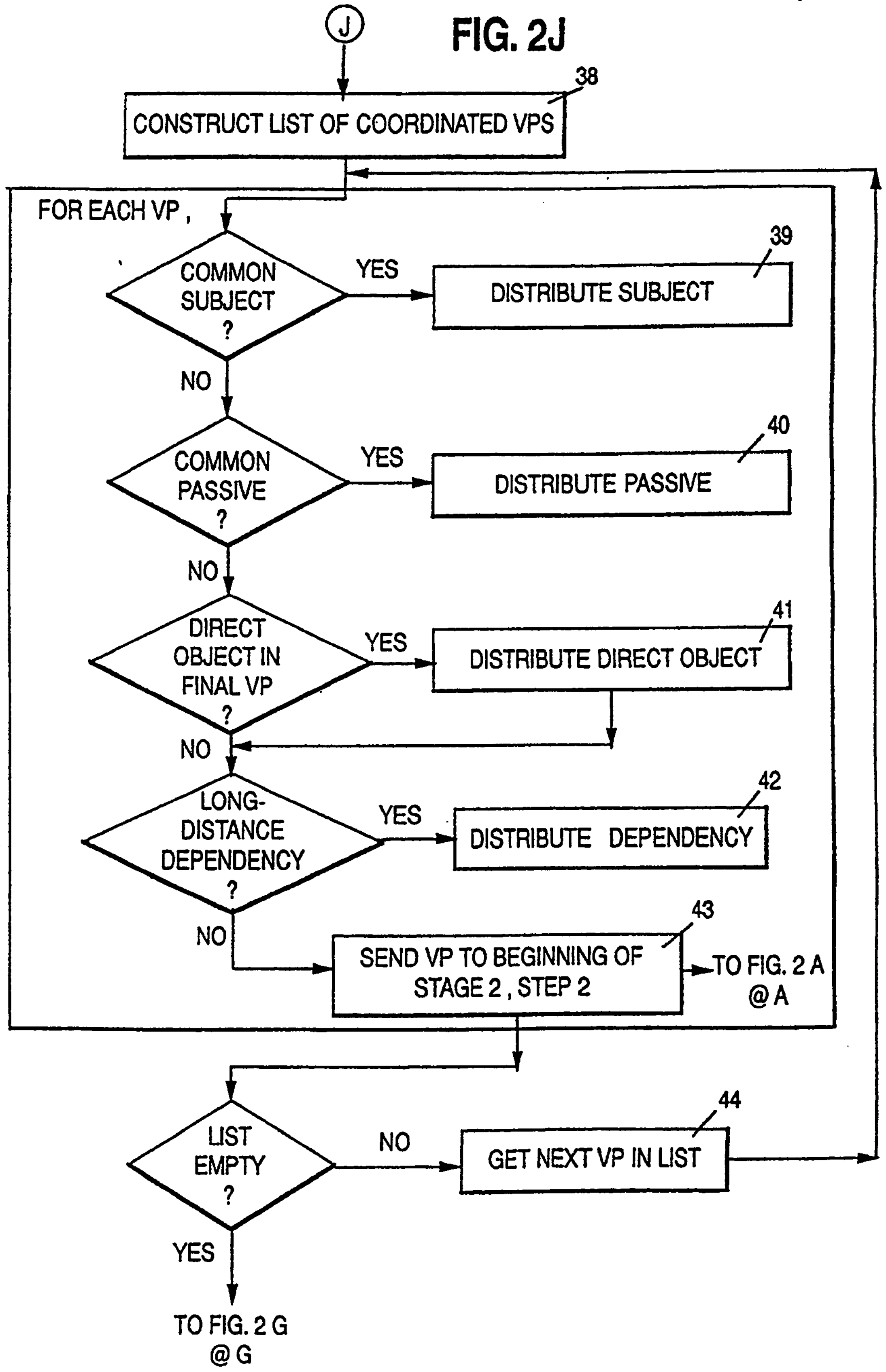

J
CONSTRUCT LIST OF COORDINATED VPS
38
FOR EACH VP ,
COMMON SUBJECT ?
YES
DISTRIBUTE SUBJECT
39
NO
COMMON PASSIVE ?
YES
DISTRIBUTE PASSIVE
40
NO
DIRECT OBJECT IN FINAL VP ?
YES
DISTRIBUTE DIRECT OBJECT
41
NO
LONG-DISTANCE DEPENDENCY ?
YES
DISTRIBUTE DEPENDENCY
42
NO
SEND VP TO BEGINNING OF STAGE 2 , STEP 2
43
TO FIG. 2 A @ A
LIST EMPTY ?
NO
GET NEXT VP IN LIST
44
YES
TO FIG. 2 G @ G

FIG. 2K

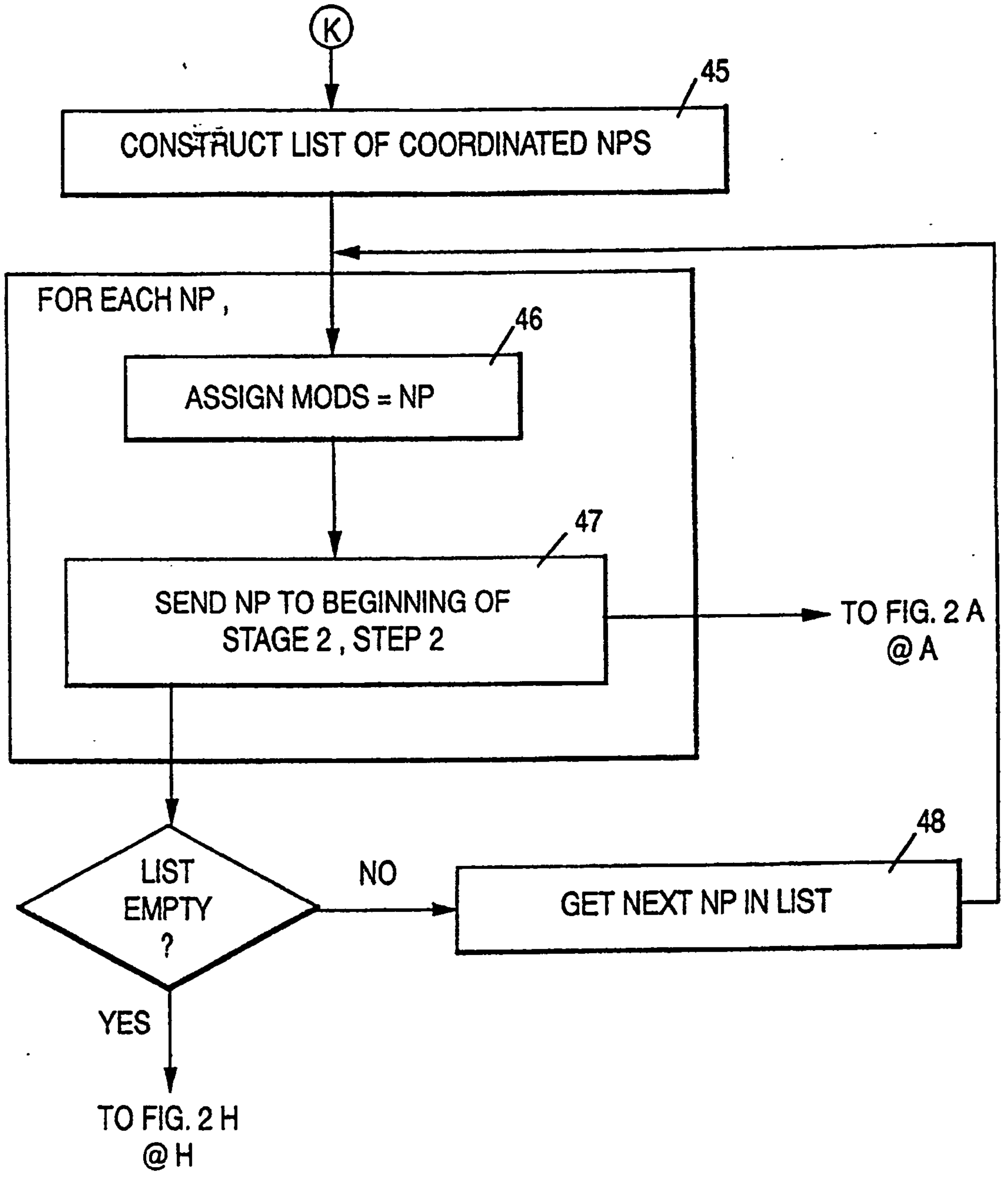

K
45
CONSTRUCT LIST OF COORDINATED NPS
FOR EACH NP ,
46
ASSIGN MODS = NP
47
SEND NP TO BEGINNING OF STAGE 2 , STEP 2
TO FIG. 2 A @ A
LIST EMPTY ?
NO
48
GET NEXT NP IN LIST
YES
TO FIG. 2 H @ H

FIG. 3

SYNTACTIC PARSE TREE FOR THE SENTENCE:

```
- - - - - - - - - - - - - - - - - - - - - - - - - -
49 — DECL1    NP1      NOUN1*     " ALICE "
                        PUNC1      " , "
               NP2      NOUN2*     " PETER "
               VERB1*   " SAID "
               VP1      COMPL1     " THAT "
                        NP3        NOUN3*      "JOHN "
                        VERB2*     " KISSED "
               PUNC2    " . "
- - - - - - - - - - - - - - - - - - - - - - - - - -
```

BASIC SEMANTIC (ARGUMENT) STRUCTURE FOR THE SENTENCE :

```
50 —  PRED        ' SAY '
      DSUBJECT    PRED        ' PETER '
      DOBJECT     PRED        ' KISS '
                  DSUBJECT    PRED        ' JOHN '
                  DOBJECT     PRED        ' ALICE '
```

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 413 132 A3**

(12) **EUROPEAN PATENT APPLICATION**

(21) Application number: **90113224.1**

(51) Int. Cl.5: **G06F 15/38**, G06F 15/20

(22) Date of filing: **11.07.90**

(30) Priority: **16.08.89 US 395118**

(43) Date of publication of application:
20.02.91 Bulletin 91/08

(84) Designated Contracting States:
DE FR GB

(88) Date of deferred publication of the search report:
31.03.93 Bulletin 93/13

(71) Applicant: **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: **Jensen, Karen**
5822 Inman Park Circle, No. 210
Rockville, MD 20852(US)

(74) Representative: **Jost, Ottokarl, Dipl.-Ing.**
IBM Deutschland Informationssysteme
GmbH Patentwesen und Urheberrecht
Pascalstrasse 100
W-7000 Stuttgart 80 (DE)

(54) A computer method for identifying predicate-argument structures in natural language text.

(57) A computer method is disclosed for determining predicate-argument structures in input prose sentences of English. The input sentence, in the form of a string of words separated by blanks, is first analyzed (parsed) by a rule component that has access only to morphological and syntactic information about the words. The output of this rule component, in the form of a data structure consisting of attribute-value pairs, is then processed by the argument-structure component, which consists of a set of partially ordered procedures that incorporate further linguistic knowledge. The output of these procedures is the same attribute-value structure, now enhanced by the presence of semantic (i.e., meaningful, non-syntactic) attributes. These semantic attributes, taken together, form the argument structure of the input sentence.

The resulting invention constitutes a fully modular, comprehensive and efficient method for passing from syntax to the first stage of semantic processing of natural (human) language. The invention applies to all prose sentences of the language for which it is designed, and not just to a subset of those sentences. It does not use domain-specific semantic information to improve the accuracy or efficiency of the syntactic component. It therefore constitutes an unrestricted broad-coverage method for natural language processing (NLP), as opposed to the restricted methods used in most NLP applications today.

Although the specific rules and procedures will be different for different natural languages, the general concept embodied in this invention is applicable to all natural languages.

FIG. 3

SYNTACTIC PARSE TREE FOR THE SENTENCE:

```
-----------------------------------------------------------
49 — DECL1   NP1       NOUN1*     'ALICE'
                       PUNC1      ','
             NP2       NOUN2*     'PETER'
             VERB1*    'SAID'
             VP1       COMPL1     'THAT'
                       NP3        NOUN3*     'JOHN'
                       VERB2*     'KISSED'
             PUNC2     '.'
-----------------------------------------------------------
```

BASIC SEMANTIC (ARGUMENT) STRUCTURE FOR THE SENTENCE :

```
50 — PRED         'SAY'
     DSUBJECT     PRED        'PETER'
     DOBJECT      PRED        'KISS'
                  DSUBJECT    PRED        'JOHN'
                  DOBJECT     PRED        'ALICE'
```

European Patent Office

EUROPEAN SEARCH REPORT

Application Number

EP 90 11 3224

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.5)
A	IBM JOURNAL OF RESEARCH AND DEVELOPMENT. vol. 32, no. 2, March 1988, NEW YORK US pages 251 - 267 , XP000022626 P. VELARDI ET AL 'Conceptual graphs for the analysis and generation of sentences' * abstract * * page 251, right column, line 13 - page 253, right column, line 9; figure 2 * ---	1-12	G06F15/38 G06F15/20
A	SYSTEMS & COMPUTERS IN JAPAN vol. 19, no. 2, February 1988, NEW YORK US pages 85 - 100 , XP000104825 Y. ITOH ET AL 'A Process of Understanding Sentences and Use of Knowledge' * page 86, left column, line 27 - page 87, left column, line 40 * ---	1-12	
A	GB-A-2 209 614 (SHARP KABUSHIKI KAISHA) 17 May 1989 * abstract * * page 6, line 3 - page 8, line 1; figure 2 * ---	1-12	
			TECHNICAL FIELDS SEARCHED (Int. Cl.5)
A	PROCEEDINGS OF THE 12TH ANNUAL INTERNATIONAL ACMSIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL. 25-28 JUNE 1989, CAMBRIDGE, US pages 117 - 126 , XP000239142 D.P. METZLER & S.W. HAAS 'The Constituent Object Parser: Syntactic Structure Matching for Information Retrieval' * page 118, right column, line 13 - line 42; figures 2,3 * -----	1,2,5,6, 9,10	G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	29 JANUARY 1993	BURNE S.R.

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 17/10, 17/27, 15/00	A2	(11) International Publication Number: WO 00/49517 (43) International Publication Date: 24 August 2000 (24.08.00)

(21) International Application Number: PCT/US00/04118

(22) International Filing Date: 18 February 2000 (18.02.00)

(30) Priority Data:
60/120,659 19 February 1999 (19.02.99) US

(71) Applicant *(for all designated States except US)*: THE TRUSTEES OF COLUMBIA UNIVERSITY IN THE CITY OF NEW YORK [US/US]; 116th Street and Broadway, New York, NY 10027 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: MCKEOWN, Kathleen, R. [US/US]; 20 Prospect Road, Wayne, NJ 07470 (US). BARZILAY, Regina [US/US]; 548 Riverside Drive, Apt. 4B, New York, NY 10027 (US).

(74) Agents: TANG, Henry et al.; Baker Botts, LLP, 30 Rockefeller Plaza, New York, NY 10112-0228 (US).

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: MULTI-DOCUMENT SUMMARIZATION SYSTEM AND METHOD

(57) Abstract

A summary for a collection of related documents can be generated by extracting phrases from the documents which include common focus elements. Phrase intersection analysis is then performed on the extracted phrases to generate a phrase intersection table, where identical or equivalent phrases are identified. Temporal processing on the phrases in the phrase intersection table is performed to remove ambiguous time references and to sort the phrases in a temporal sequence. Sentence generation is then used to combine the phrases in the phrase intersection table into a coherent summary.

MULTI-DOCUMENT SUMMARIZATION SYSTEM AND METHOD

SPECIFICATION

Statement of Government Rights

The United States Government may have certain rights to the invention set forth herein pursuant to a grant by the National Science Foundation, Contract No. IRI-96-18797.

Statement of Related Applications

This application claims the benefit of United States provisional patent application, Serial No. 60/120,659, entitled "Information Fusion in the Context of Multi-Document Summarization," which was filed on February 19, 1999.

Field of the Invention

The present invention relates generally to information summarization and more particularly relates to systems and methods for generating a summary for a set of multiple, related documents.

Background of the Invention

The amount of information available today drastically exceeds that of any time in history. With the continuing expansion of the Internet, this trend will likely continue well into the future. Often, people conducting research of a topic are faced with information overload as the number of potentially relevant documents exceeds the researchers ability to individually review each document. To address this problem, information summaries are often relied on by researchers to quickly evaluate a document to determine if it is truly relevant to the problem at hand.

Given the vast collection of documents available, there is interest in developing and improving the systems and methods used to summarize information content. For individual documents, domain-dependent template based systems and domain-independent sentence extraction methods are known. Such known systems can provide a reasonable summary of a single document. However, these systems are not able to compare and contrast related documents in a document set to provide a summary of the collection.

The ability to summarize collections of documents containing related information is desirable to further expedite the research process. For example, for a researcher interested in news stories regarding a certain event, a summary of all documents from a given source, or multiple sources, would provide a valuable overview of the documents within the set. From such a summary, the researcher may be able to extract the information desired, or at the very least, make an informed decision regarding the relevance of the set of documents. Therefore, there remains a need for systems and methods which can generate a summary of related documents in a document set.

Summary of the Invention

It is an object of the present invention to provide a system and method for generating a summary of a set of multiple, related documents.

It is a further object of the present invention to provide a system and method for generating a summary of a set of multiple, related documents which use paraphrasing rules to detect similarities in non-identical phrases in the documents.

A present method for generating a summary of related documents in a collection includes extracting phrases from the documents which have common focus elements. Phrase intersection analysis is performed on the extracted phrases to generate a phrase intersection table. Temporal processing can be performed on the phrases in the phrase intersection table to remove ambiguous temporal references and to sort the phrases in a temporal sequence. Sentence generation is performed using the phrases in the phrase intersection table to generate the multidocument summary.

Preferably, the phrase intersection analysis operation can include representing the phrases in tree structures having root nodes and children nodes; selecting those tree structures with verb root nodes; comparing the selected root nodes to the other root nodes to identify identical nodes; applying paraphrasing rules to non-identical root nodes to determine if non identical nodes are equivalent; and evaluating the children nodes of those tree structures where the parent nodes are identical or equivalent. The tree structure can take the form of a DSYNT tree structure. The paraphrasing rules can include one or more rules which are selected from the group consisting of ordering of sentence components, main clause versus a relative clause, different syntactic categories, change in grammatical features, omission of an empty head, transformation of one part of speech to another, and semantically related words.

In an embodiment of the present method, the temporal processing includes time stamping phrases based on a first occurrence of the phrase in the collection; substituting date certain references for ambiguous temporal references; ordering the phrases based on the time stamp; and inserting a temporal marker if a temporal gap between phrases exceeds a threshold value.

Preferably, a phrase divergence processing operation can also be performed to include phrases that signal changes in focus of the documents in the collection.

Sentence generation can includes mapping the phrases, represented in the tree structure, to an input format of a language generation engine and then operating the language generation engine.

A present system for generating a summary of a collection of related documents includes a storage device for storing the documents in the collection, a lexical database, such as WordNet, and a processing subsystem operatively coupled to the storage device and the lexical database. The processing subsystem is programmed to perform multiple document summarization including: accessing the documents in the storage device; using the lexical database to extract phrases from the documents with similar focus elements; performing phrase intersection analysis on the extracted phrases to generate a phrase intersection table; performing temporal processing on the phrases in the phrase intersection table; and performing sentence generation using the phrases in the phrase intersection table.

The methods described above can be encoded in the form of a computer program stored in computer readable media, such as CD-ROM, magnetic storage and the like.

Brief Description of the Drawing

Further objects, features and advantages of the invention will become apparent from the following detailed description taken in conjunction with the accompanying figures showing illustrative embodiments of the invention, in which

Figure 1 is a flow chart illustrating the operation of the present multiple document summarization system;

Figure 2 is a flow chart of a phrase intersection processing operation in accordance with the system operation of Figure 1;

Figure 3 is a pictorial diagram of a DSYNT tree structure for an exemplary sentence;

Figure 4 is a flow chart of a temporal processing operation in accordance with the system operation of Figure 1;

Figure 5 is a simplified block diagram of an embodiment of the present multiple document summarization system.

Throughout the figures, the same reference numerals and characters, unless otherwise stated, are used to denote like features, elements, components or portions of the illustrated embodiments. Moreover, while the subject invention will now be described in detail with reference to the figures, it is done so in connection with the illustrative embodiments. It is intended that changes and modifications can be made to the described embodiments without departing from the true scope and spirit of the subject invention as defined by the appended claims.

Detailed Description of Preferred Embodiments

Figure 1 is a flow chart which provides an overview of the operation of the present multiple document summarization system. Initially, a set of documents, in computer readable format and grouped by a common theme or domain, is presented to the summarization system. From the collection of documents, entities are identified and sentences are extracted from the documents which are relevant to the focus of the articles. Entities can be identified and extracted in a number of ways, such as by use of an information extraction engine. A suitable information extraction engine is TALENT, which is available from International Business Machines, Inc. In step 100, phrases are extracted from the documents which include terms that are present in at least two of the documents. In addition, divergent phrases, which may be indicative of contrasts in the documents, are also extracted from the document in step 110. Following extraction, phrase intersection processing (step 120) and phrase divergence processing (step 130) are performed to evaluate and compare the extracted phrases and determine whether such phrases should be included in the resulting multiple document summary. Since phrases are extracted from multiple documents and can include temporal references which are ambiguous when taken out of context from the original document, temporal processing (step 140) is performed on the phrases selected for the summary. Finally, sentence generation (step 150) is used to transform selected phrases into a coherent summary.

Figure 2 is a flow chart which further illustrates steps that can be performed in connection with phrase intersection processing of step 120. The selected phrases are grammatically parsed and represented in a tree structure, such as a DSYNT tree diagram, which is generally known in the art. An example of such a diagram is illustrated in Figure 3. The parse trees can be generated by a conventional grammatical parser, such as Collin's parser. The DSYNT tree structure is a way of representing the constituent dependencies resulting from a predicate-argument sentence structure. In the tree structure, each non-auxiliary word in the sentence has a node which is connected to its direct dependents. Grammatical features of each word are also stored in the node. To facilitate subsequent comparisons, words in the nodes are kept in their canonical form.

Returning to Figure 2, those trees which have root nodes which are verbs are selected and used as the basis for comparison. Each such verb based tree is compared against the other trees derived from the sentences extracted from the documents in the collection (step 220). A comparison is made to determine if two nodes are identical (step 230). If two nodes are identical, those nodes are added to an output tree (step 235) and the nodes are evaluated to determine if there are further nodes descending from the root node (step 240). Such further nodes are referred to as children nodes. If children nodes are present (step 245), the comparison in step 230 is repeated for each of child node. If the analysis of the children nodes is complete at step 240, a determination is made as to whether the trees with common root nodes represent a phrase intersection (step 250). For example, if there is commonality in the root node and at least two children nodes of that root node, that phrase can be deemed complete and added to a phrase intersection table (step 255). If no phrase intersection is detected at step 250, the next parent node is selected for processing (step 260) and control returns to step 230.

Returning to step 230, if two nodes are not identical, it is still possible for the nodes to be equivalent. To make this determination, the present method employs a set of paraphrasing rules to evaluate the nodes (step 265). Paraphrasing, which can be broadly defined as alternative ways a human speaker can choose to "say the same thing" by using linguistic knowledge, generally occurs at a "surface" level, e.g., it is achieved by using semantically related words and syntactic transformations.

In the case of a set of related documents, theme sentences of the documents will generally be close semantically. This limits the scope of different paraphrasing types to be evaluated. From an analysis of paraphrasing patterns evaluated through themes of a training corpus derived from TDT, the following non-exhaustive set of paraphrasing categories have been found to occur with the greatest frequency:

1. ordering of sentence components: "*Tuesday they met...*" and "*They met ... Tuesday*";
2. main clause vs. a relative clause: "*...a building was devastated by the bomb*" and "*...a building, devastated by the bomb*";
3. realization in different syntactic categories, e.g., classifier vs. apposition: "*Palestinian leader Arafat*" and "*Arafat, Palestinian leader*", *"Pentagon speaker" and "speaker from the Pentagon*";
4. change in grammatical features: active/passive, time, number. "*...a building was devastated by the bomb*" and "*...the bomb devastated a building*";
5. omission of an empty head: "*group of students*" and "*students*";
6. transformation from one part of speech to another: "*building devastation*" and "*...building was devastated*"; and

7. using semantically related words such as synonyms: "*return*" and "*alight*", "*regime*" and "*government*".

The categories presented are used as paraphrasing rules by the present methods. The majority of these categories,such as ordering, can be fully implemented in an automatic way, . However, some of the rules can be only approximated to a certain degree in an automated system. For example, identification of similarity based on semantic relations between words depends on the scope of coverage of the thesaurus employed. Word similarity can be established using relationships such as synonymy, hyponymy/hypernymy, and meronymy/holonymy which are detectable using the WordNet language database which is described in the article "WordNet: A lexical Database for English", by G.A. Miller, Communications of the ACM, Vol. 38, No. 11. pp. 39-41, November 1995.

If any of the included paraphrasing rules are satisfied for non-identical nodes, the nodes are deemed equivalent (step 270). Equivalent nodes are added to the output tree (step 235) and processed in the same manner as identical nodes. If no paraphrasing rule is applicable to non-identical nodes, there is no phrase intersection with the current tree (step 280).

In addition to phase intersection processing, which compares phrases for similarity, it is also desirable to perform phrase divergence processing (step 130), which compares selected phrases for differences. Phrase divergence may indicate a critical change in the course of events through a set of related documents and would be worthy of inclusion in a summary. For example, a collection of articles regarding a plane crash could begin with a focus on the passengers as “survivors” and later refer to “casualties,” “victims,” “bodies” and the like, which signify a turning point in the events described by the documents. WordNet can also be used in phrase divergence processing by evaluating focus relationships such antonymy (e.g., "happiness is opposite to sadness").

Once phrases are selected from the documents for the summary, temporal processing can be performed to sequence the phrases and eliminate ambiguous temporal references. The flow chart of Figure 4 illustrates an overview of the temporal processing operations performed in the present methods. Using a rule that an event is assumed to have occurred on the day that it is first reported, a time stamp can be applied to the selected phrases based on the earliest occurrence of the phrase in the collection of documents (step 405). In certain cases, phrases may include ambiguous temporal references, such as today, yesterday, etc. In this case, such ambiguous references can be replaced by a date certain reference, such as by changing "Yesterday it was reported...." to "*On 01/02/2000*, it was reported...". Such substitutions, which are performed in step 410, can be implemented using the Emacs "calendar" package.

The extracted phrases can then be ordered in accordance with the assigned date stamp (step 415). In certain cases, a large temporal gap may exist between consecutive phrases. In such a case, if the gap exceeds a threshold, such as two days, a temporal marker can be inserted between the phrases to indicate this gap in time (step 420). This may be significant, for example, in the case of a collection of news articles where the gap in time can also correspond to a change in focus in the articles.

With the phrases selected and sorted in temporal order, sentence generation (step 150) can be performed to synthesize a coherent summary. Sentence generation involves two major operations. First, the DSYNT representation of the phrases to be used in sentence generation are mapped to the appropriate syntax of a selected language generation engine. Then, the language generation engine is operated to arrange the phrases into coherent sentences. A suitable language generation engine is FUF/SURGE, which is available from Columbia University, New York , New York, as well as from Ben Gurion University, Department of Computer Science, Beer-Sheva, Israel. The acronym FUF stands for Functional Unification Formalism interpreter and the acronym SURGE stands for syntactic realization grammar for text generation. The input specification for the FUF/SURGE engine includes a semantic role, *circumstantial,* which itself includes a temporal feature. The inclusion of the semantic attributes enables FUF/SURGE to perform various paraphrasing operations to the input phrases to improve the resulting sentences.

Figure 5 is a simplified block diagram of a multiple document summarization system in accordance with the present invention. The system 500 includes a processor section 505 wherein the processing operations set forth in Figure 1 are performed. The system also includes non-volatile storage coupled to the processor section 505 for document storage 510, collection summary storage 515, lexical database storage 518 and program storage 520. Generally these storage systems are read/write data storage systems, such as magnetic media and read/write optical storage media. However, the document collection storage may take the form of read-only storage, such as a CD-ROM storage device. The system further includes RAM memory 525 coupled to the processor section for temporary storage during operation. The system 500 will generally include one or more input device 530 such as a keyboard, digitizer, mouse and the like, which is coupled to the processor section 505. Similarly, a conventional display device 535 is generally provided which is also operatively coupled to the processor section.

The particular hardware embodiment is not critical to the practice of the present invention. Various computer platforms and architectures can be used to implement the system 500, such as personal computers, workstations, networked computers, and the like. The functions described in the system can be performed locally or in a distributed manner, such as over a local area network or the Internet. For example, the document collection storage 510 may be at a remote archive location which is accessed by the processor section 505 via a connection to the Internet.

Although the present invention has been described in connection with specific exemplary embodiments, it should be understood that various changes, substitutions and alterations can be made to the disclosed embodiments without departing from the spirit and scope of the invention as set forth in the appended claims.

CLAIMS

1. A method for generating a summary of a plurality of related documents in a collection comprising:

extracting phrases having focus elements from the plurality of documents;

performing phrase intersection analysis on the extracted phrases to generate a phrase intersection table;

performing temporal processing on the phrases in the phrase intersection table; and

performing sentence generation using the phrases in the phrase intersection table.

2. The method of generating a summary as defined by claim 1, wherein the phrase intersection analysis comprises:

representing the phrases in tree structures having root nodes and children nodes;

selecting those tree structures with verb root nodes;

comparing the selected root nodes to the other root nodes to identify identical nodes;

applying paraphrasing rules to non-identical root nodes to determine if non identical nodes are equivalent; and

evaluating the children nodes of those tree structures where the parent nodes are identical or equivalent.

3. The method of claim 2, wherein the tree structure is a DSYNT tree structure.

4. The method of claim 2, wherein the paraphrasing rules are selected from the group consisting of ordering of sentence components, main clause versus a relative clause, different syntactic categories, change in grammatical features, omission of an

empty head, transformation of one part of speech to another, and semantically related words.

5. The method of claim 1, wherein the temporal processing includes:

time stamping phrases based on a first occurrence of the phrase in the collection;

substituting date certain references for ambiguous temporal references;

ordering the phrases based on the time stamp; and

inserting a temporal marker if a temporal gap between phrases exceeds a threshold value.

6. The method of claim 1, further comprising a phrase divergence processing operation.

7. The method of claim 1, wherein the sentence generation includes mapping phrases to an input format of a language generation engine and operating the language generation engine.

8. A system for generating a summary of a plurality of related documents in a collection comprising:

a storage device for storing the documents in the collection;

a lexical database; and

a processing subsystem, the processing subsystem being operatively coupled to the storage device and the lexical database, the processing subsystem being programmed to access the documents in the storage device and:

using the lexical database to extract phrases having focus elements from the plurality of documents;

performing phrase intersection analysis on the extracted phrases to generate a phrase intersection table;

performing temporal processing on the phrases in the phrase intersection table; and performing sentence generation using the phrases in the phrase intersection table.

9. The system for generating a summary as defined by claim 9, wherein the phrase intersection analysis processing further comprises:

representing the phrases as data structures having root nodes and children nodes;

selecting those data structures with verb root nodes;

comparing the selected root nodes to the other root nodes to identify identical nodes;

applying paraphrasing rules to non-identical root nodes to determine if non identical nodes are equivalent; and

evaluating the children nodes of those tree structures where the parent nodes are identical or equivalent.

10. The system of claim 9, wherein the data structure is a DSYNT tree structure.

11. The system of claim 9, wherein the paraphrasing rules are selected from the group consisting of ordering of sentence components, main clause versus a relative clause, different syntactic categories, change in grammatical features, omission of an empty head, transformation of one part of speech to another, and semantically related words.

12. The system of claim 8, wherein the temporal processing includes:

time stamping phrases based on a first occurrence of the phrase in the collection;

substituting date certain references for ambiguous temporal references;

ordering the phrases based on the time stamp; and

inserting a temporal marker if a temporal gap between phrases exceeds a threshold value.

13. The system of claim 8, further comprising a phrase divergence processing operation.

14. The system of claim 8, wherein the processing subsystem includes a language generation engine and wherein sentence generation includes mapping phrases to an input format of the language generation engine and then operating the language generation engine.

15. The system of claim 8, wherein the storage device for storing the documents in the collection is remotely located from the processing subsystem.

16. A computer readable media for programming a computer system to perform a method of generating a summary of a plurality of related documents in a collection comprising:

extracting phrases having focus elements from the plurality of documents;

performing phrase intersection analysis on the extracted phrases to generate a phrase intersection table;

performing temporal processing on the phrases in the phrase intersection table; and

performing sentence generation using the phrases in the phrase intersection table.

17. The computer readable media of claim 16, wherein the phrase intersection analysis comprises:

representing the phrases in tree structures having root nodes and children nodes;

selecting those tree structures with verb root nodes;

comparing the selected root nodes to the other root nodes to identify identical nodes;

applying paraphrasing rules to non-identical root nodes to determine if non identical nodes are equivalent; and evaluating the children nodes of those tree structures where the parent nodes are identical or equivalent.

18. The computer readable media of claim 17, wherein the tree structure is a DSYNT tree structure.

19. The computer readable media of claim 17, wherein the paraphrasing rules are selected from the group consisting of ordering of sentence components, main clause versus a relative clause, different syntactic categories, change in grammatical features, omission of an empty head, transformation of one part of speech to another, and semantically related words.

20. The computer readable media of claim 16, wherein the temporal processing includes:

time stamping phrases based on a first occurrence of the phrase in the collection;

substituting date certain references for ambiguous temporal references;

ordering the phrases based on the time stamp; and

inserting a temporal marker if a temporal gap between phrases exceeds a threshold value.

21. The computer readable media of claim 16, further comprising a phrase divergence processing operation.

22. The computer readable media of claim 16, wherein the sentence generation includes mapping phrases to an input format of a language generation engine and operating the language generation engine.

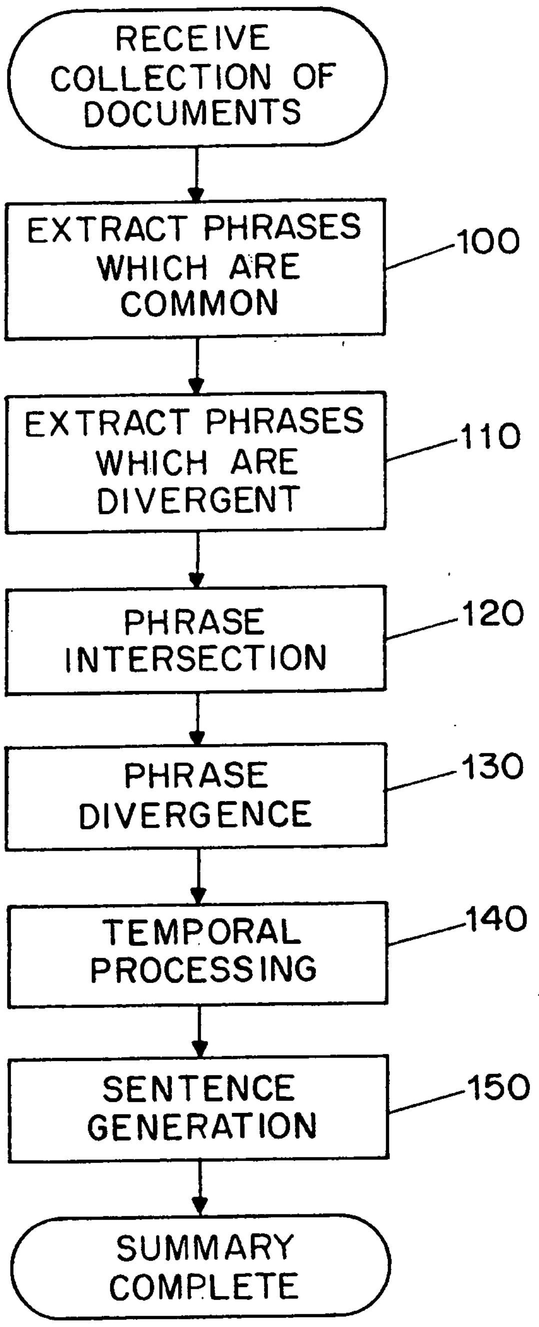

RECEIVE COLLECTION OF DOCUMENTS
EXTRACT PHRASES WHICH ARE COMMON
100
EXTRACT PHRASES WHICH ARE DIVERGENT
110
PHRASE INTERSECTION
120
PHRASE DIVERGENCE
130
TEMPORAL PROCESSING
140
SENTENCE GENERATION
150
SUMMARY COMPLETE

FIG. 1

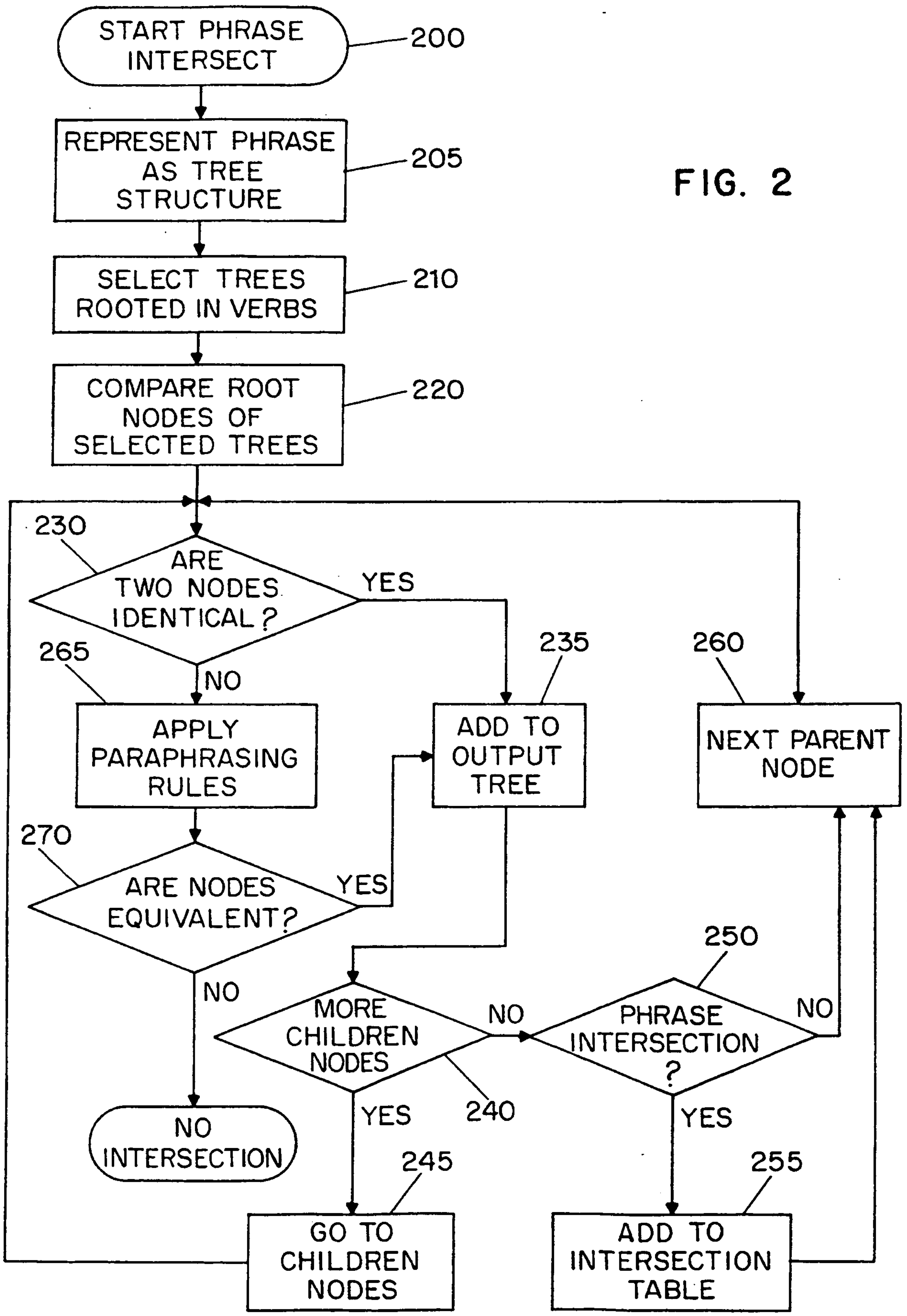

START PHRASE INTERSECT
200
REPRESENT PHRASE AS TREE STRUCTURE
205
SELECT TREES ROOTED IN VERBS
210
COMPARE ROOT NODES OF SELECTED TREES
220
230
ARE TWO NODES IDENTICAL?
YES
NO
235
ADD TO OUTPUT TREE
260
NEXT PARENT NODE
265
APPLY PARAPHRASING RULES
270
ARE NODES EQUIVALENT?
YES
NO
NO INTERSECTION
MORE CHILDREN NODES
240
NO
YES
250
PHRASE INTERSECTION ?
NO
YES
245
GO TO CHILDREN NODES
255
ADD TO INTERSECTION TABLE

FIG. 2

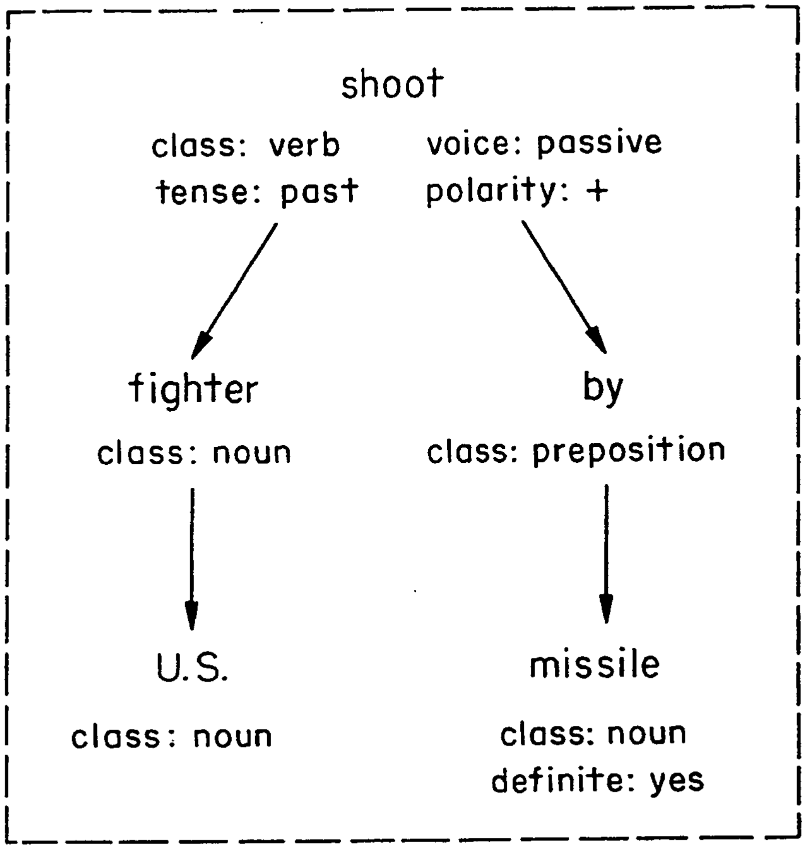

shoot
class: verb
tense: past
voice: passive
polarity: +
fighter
class: noun
by
class: preposition
U.S.
class: noun
missile
class: noun
definite: yes

FIG. 3

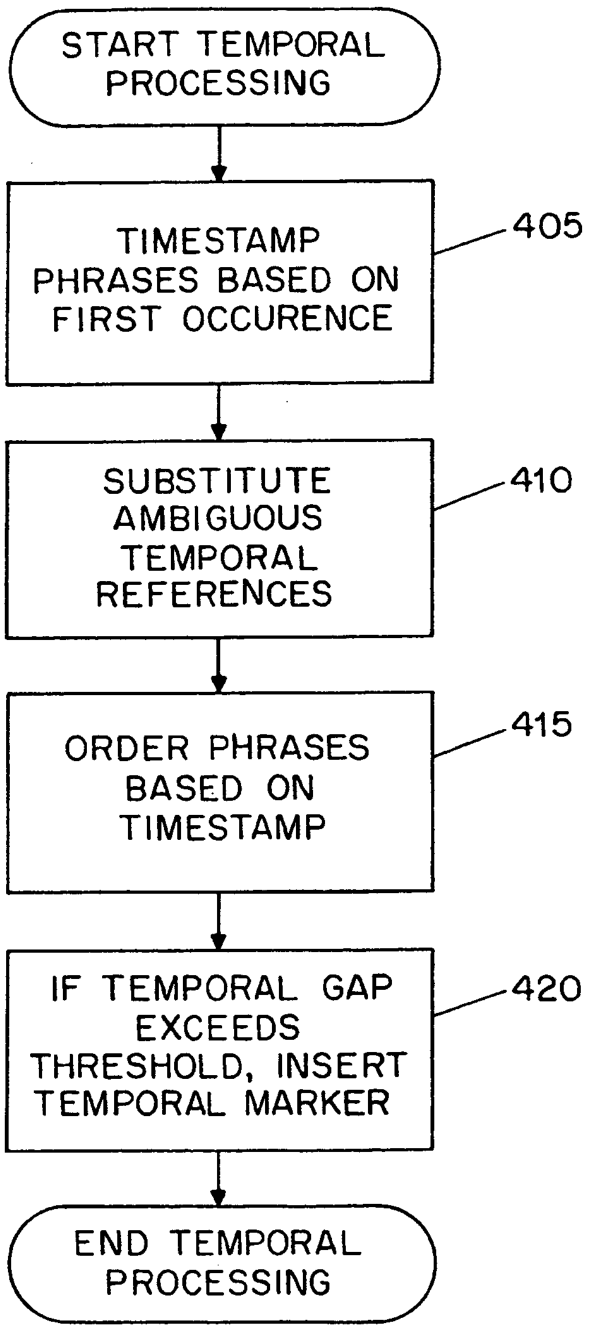

START TEMPORAL PROCESSING
TIMESTAMP PHRASES BASED ON FIRST OCCURENCE
405
SUBSTITUTE AMBIGUOUS TEMPORAL REFERENCES
410
ORDER PHRASES BASED ON TIMESTAMP
415
IF TEMPORAL GAP EXCEEDS THRESHOLD, INSERT TEMPORAL MARKER
420
END TEMPORAL PROCESSING

FIG. 4

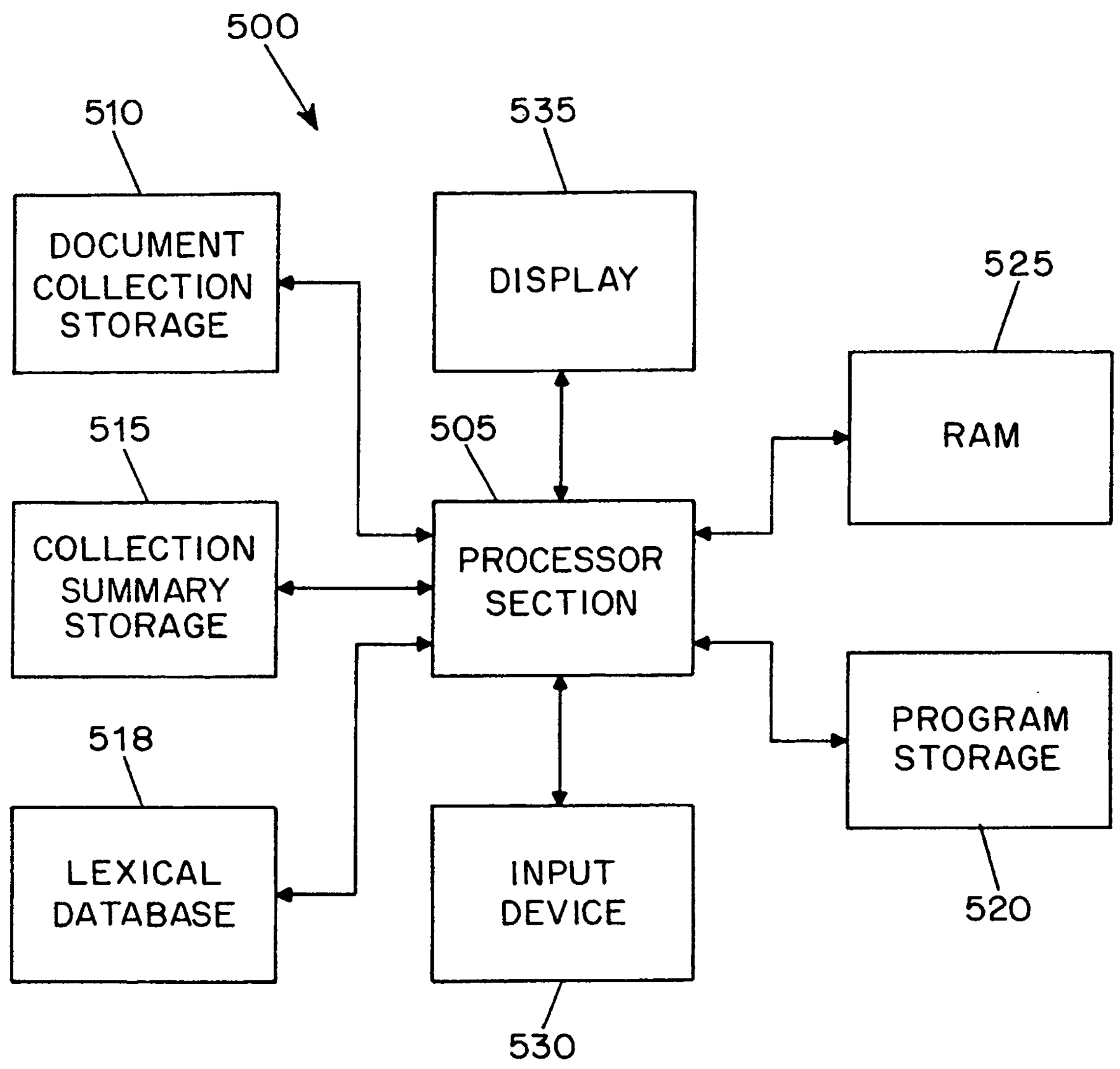

500
510
DOCUMENT COLLECTION STORAGE
535
DISPLAY
525
RAM
515
COLLECTION SUMMARY STORAGE
505
PROCESSOR SECTION
518
LEXICAL DATABASE
INPUT DEVICE
530
PROGRAM STORAGE
520

FIG. 5